babz

Finanzsoftware

Benutzerhandbuch

Rechenkern Version 2.0

Versionsinformationen

Stand des Handbuchs:	September 2012
Versionsnummer des Rechenkerns:	2.0.800

Rechtliche Hinweise

Die Nutzung und Weitergabe dieses Dokuments unterliegt den „babz Nutzungsbedingungen“ in der aktuellen Version.

Windows®, Excel®, Visual Basic® und ActiveX® sind eingetragene Warenzeichen der Microsoft Corporation

Kontakt

Marc Speicher
Südring 77
69514 Laudenbach

email: marc@babz-software.de
Internet: http://www.babz-software.de

Inhaltsübersicht

Inhaltsübersicht I
Inhaltsverzeichnis II
1 Einführung 10
2 Klasse BASystem 18
3 Klasse BACalendar 28
4 Klasse BAInterestCurve 41
5 Klasse BAFXForwardCurve 61
6 Klasse BAOutput 71
7 Klasse BALoan 80
8 Klasse BAInterestSwap 100
9 Klasse BAFXForward 110
10 Klasse BAPrepaymentCompensation 118
11 Klasse BALoanPricing 127
12 Klasse BAAssistant 134
13 Interface für Microsoft Excel® 138
Anhang I Laufzeitfehler 143
Anhang II Registrierung der Datei babz.dll 153
Anhang III Nutzungsbedingungen 157
Anhang IV Historie 160

Inhaltsverzeichnis

Inhaltsübersicht I
Inhaltsverzeichnis II
1 Einführung 10
1.1 Zielsetzung und Anwendungsbereich 10
1.2 Technik & Aufbau 12
1.3 Fehlerbeseitigung und Optimierung 15
1.4 Ausblick 15
1.5 Hinweise zum Handbuch 16
2 Klasse BASystem 18
2.1 Aufgaben & Anwendung 18
2.2 Eigenschaften 18
2.2.1 AccruedInterestSettlementDate 18
2.2.2 CashflowsStartDate 19
2.2.3 CurrRequired 20
2.2.4 DigitsDiscountFactor 20
2.2.5 DigitsInterestRate 21
2.2.6 InterestCurveUserExit 21
2.2.7 NoRegKeyToRegistry 22
2.2.8 NoRegKeyUserDialog 22
2.2.9 RegKey 22
2.2.10 Trace 23
2.2.11 TraceCache 24
2.2.12 TraceLevel 25
2.3 Methoden 25
2.3.1 ClearTrace 25
2.3.2 LoadRegKey 26
2.3.3 SetRegKey 26
2.3.4 WriteTraceToFile 26
3 Klasse BACalendar 28
3.1 Aufgaben & Anwendung 28
3.2 Eigenschaften 28
3.2.1 Description 28
3.2.2 Calendars 28
3.2.3 Holidays 29
3.2.4 Tag 34

3.3 Methoden 34
3.3.1 AddDateInverval 34
3.3.2 AddPredefinedHoliday 35
3.3.3 Clear 35
3.3.4 GetDateFromBusinessDays 35
3.3.5 GetDayCountFraction 36
3.3.6 GetInterestDays 37
3.3.7 GetYearInterestDays 37
3.3.8 IsBankHoliday 37
3.3.9 Load 38
3.3.10 SetDay 39
3.3.11 ShiftDate 40
4 Klasse BAInterestCurve 41
4.1 Aufgaben & Anwendung 41
4.2 Eigenschaften 41
4.2.1 BusinessDayConvention_CashRates 41
4.2.2 BusinessDayConvention_SwapRates 41
4.2.3 BusinessDayConvention_ZeroRates 42
4.2.4 Calendar 42
4.2.5 Curr 43
4.2.6 Description 43
4.2.7 DayCountConvention_CashRates 43
4.2.8 DayCountConvention_SwapRates 44
4.2.9 DayCountConvention_ZeroRates 44
4.2.10 DiscountFactors 45
4.2.11 InterestRates 46
4.2.12 Interpolation_Bootstrapping 48
4.2.13 Interpolation_DiscountFactors 48
4.2.14 Fixing 49
4.2.15 LastDate 49
4.2.16 Calculated 49
4.2.17 Settlement 50
4.2.18 SettlementDays 50
4.2.19 Spread 50
4.2.20 SwapFrequency 51
4.2.21 SwapInterestMethod 51
4.2.22 Tag 52

4.2.23 UserExitObject 52
4.3 Methoden 54
4.3.1 Clone 54
4.3.2 GetDF 54
4.3.3 GetForwardCurve 55
4.3.4 GetForwardRate 55
Formeln & Berechnungsgrundlagen 55
4.3.5 GetLastDate 56
4.3.6 Calculate 56
Formeln & Berechnungsgrundlagen 57
Exkurs: Ermittlung von Diskontfaktoren aus Swapsätzen 58
Hinweise zur Verwendung von Marktdaten 59
4.4 Ereignisse 60
4.4.1 AfterCalculation 60
4.4.2 PropertyChanged 60
5 Klasse BAFXForwardCurve 61
5.1 Aufgaben & Anwendung 61
5.2 Eigenschaften 61
5.2.1 AskSpot 61
5.2.2 BusinessDayConvention 61
5.2.3 BidSpot 62
5.2.4 Calendar 62
5.2.5 CurrDomestic 63
5.2.6 CurrForeign 63
5.2.7 Description 63
5.2.8 Fixing 64
5.2.9 Interpolation 64
5.2.10 LastDate 64
5.2.11 MidSpot 65
5.2.12 Outrights 65
5.2.13 Calculated 66
5.2.14 Settlement 66
5.2.15 SettlementDays 66
5.2.16 Swaps 67
5.2.17 Tag 68
5.3 Methoden 68
5.3.1 GetOutright 68

5.3.2 Calculate 69
5.4 Ereignisse 69
5.4.1 AfterCalculation 69
5.4.2 PropertyChanged 70
6 Klasse BAOutput 71
6.1 Aufgaben und Anwendung 71
6.2 Eigenschaften 71
6.2.1 Description 71
6.2.2 DataArray 71
6.2.3 DateSum 72
6.2.4 InterestCurve 75
6.2.5 PresentValue 75
6.2.6 Tag 76
6.3 Methoden 76
6.3.1 ConvertPresentValue 76
6.3.2 GetColumn 76
6.3.3 GetDateSum 77
6.3.4 GetPresentValue 77
6.3.5 GetRow 77
6.3.6 Merge 78
7 Klasse BALoan 80
7.1 Aufgaben & Anwendung 80
7.2 Übergeordnete Datentypen 80
7.2.1 baDateControl 80
7.3 Eigenschaften 81
7.3.1 Amortisation 81
7.3.2 Amount 84
7.3.3 Calculated 84
7.3.4 Calendar 85
7.3.5 Curr 85
7.3.6 Description 86
7.3.7 Disagio 86
7.3.8 Drawing 86
7.3.9 Interest 87
7.3.10 ManualAmortisation 93
7.3.11 ManualInterest 94
7.3.12 Output 96

7.3.13 Remainder 97
7.3.14 Tag 98
7.4 Methoden 98
7.4.1 Calculate 98
7.4.2 Clone 98
7.4.3 GetAccruedInterest 99
7.5 Ereignisse 99
7.5.1 AfterCalculation 99
7.5.2 PropertyChanged 99
8 Klasse BAInterestSwap 100
8.1 Aufgaben & Anwendung 100
8.2 Eigenschaften 100
8.2.1 AccruedInterest 100
8.2.2 AutoFixedLeg 100
8.2.3 Calculated 101
8.2.4 CleanPrice 102
8.2.5 Description 102
8.2.6 DirtyPrice 102
8.2.7 FixedRate 103
8.2.8 InterestCurve_Payer 103
8.2.9 InterestCurve_Receiver 104
8.2.10 PVFixed 104
8.2.11 PVFloating 104
8.2.12 Spread 105
8.2.13 SwapType 105
8.2.14 Tag 106
8.2.15 ValuationMode 106
8.3 Methoden 107
8.3.1 GetFairFixedRate 107
8.3.2 GetFairSpread 107
8.3.3 Price 108
Formeln & Berechnungsgrundlagen 108
8.4 Ereignisse 109
8.4.1 AfterCalculation 109
8.4.2 PropertyChanged 109
9 Klasse BAFXForward 110
9.1 Aufgaben & Anwendung 110

9.2 Eigenschaften 110
9.2.1 Calculated 110
9.2.2 Contracts 110
9.2.3 Description 112
9.2.4 ErrorContract 112
9.2.5 FXForwardCurve 113
9.2.6 InterestCurve 113
9.2.7 Result 113
9.2.8 Tag 114
9.2.9 ValuationMode 114
9.3 Methoden 115
9.3.1 Price 115
9.4 Ereignisse 116
9.4.1 AfterCalculation 116
9.4.2 PropertyChanged 117
10 Klasse BAPrepaymentCompensation 118
10.1 Aufgaben & Anwendung 118
10.2 Eigenschaften 118
10.2.1 Description 118
10.2.2 Compensation 118
10.2.3 CostOfAdministration 119
10.2.4 CostOfHandling 120
10.2.5 CostOfRisk 120
10.2.6 CurrentBalance 121
10.2.7 FixedUntil 121
10.2.8 InterestCurve 122
10.2.9 Loan 122
10.2.10 LoanNoRefresh 122
10.2.11 PVCashflow 123
10.2.12 PVCostOfAdministration 124
10.2.13 PVCostOfRisk 124
10.2.14 PVLostMargin 125
10.2.15 Tag 125
10.3 Methoden 125
10.3.1 Calculate 125
10.4 Ereignisse 126
10.4.1 AfterCalculation 126

10.4.2 PropertyChanged 126
11 Klasse BALoanPricing 127
11.1 Aufgaben & Anwendung 127
11.2 Eigenschaften 127
11.2.1 CalculationMode 127
11.2.2 Description 128
11.2.3 InterestCurve 128
11.2.4 InterestRate 129
11.2.5 IterationChange 129
11.2.6 IterationCount 130
11.2.7 MarginAdjustment 130
11.2.8 PresentValue 131
11.2.9 ResultLoan 131
11.2.10 ResultWeightedAmortisation 132
11.2.11 Tag 132
11.3 Methoden 133
11.3.1 Calculate 133
11.4 Ereignisse 133
11.4.1 AfterCalculation 133
11.4.2 PropertyChanged 133
12 Klasse BAAssistant 134
12.1 Aufgaben & Anwendung 134
12.2 Eigenschaften 134
12.3 Methoden 134
12.3.1 CubicSpline 134
12.3.2 GetConstantDescription 135
12.3.3 LinearInterpolation 137
13 Interface für Microsoft Excel® 138
13.1 Allgemeines 138
13.2 Einrichtung 138
13.3 Aufbau 140
13.4 Bedienung 140
Anhang I Laufzeitfehler 143
Anhang II Registrierung der Datei babz.dll 153
Anhang III Nutzungsbedingungen 157
Anhang IV Historie 160

1 Einführung

1.1 Zielsetzung und Anwendungsbereich

Wer sich im Rahmen des Corporate Treasury mit Finanzierungsfragen und dem Management von Zins- und Währungsrisiken befasst, sah sich bei der ein oder anderen Aufgabe sicherlich schon mit dem Problem konfrontiert, dass keine Software zur Verfügung stand, mit der die Bearbeitung dieser Aufgabe möglich gewesen wäre. Das kann der Fall sein, wenn man mit einem Tabellenkalkulations-Programm arbeitet, aber auch, wenn man eine spezialisierte Treasury-Software im Einsatz hat:

- Mit den Bordmitteln der verfügbaren Tabellenkalkulation allein kann die Aufgabe nicht, nur mit großem Aufwand oder nur sehr ungenau gelöst werden.
- Das installierte Treasury-Management-System ist für die konkrete Aufgabenstellung nicht geeignet, müsste erst konfiguriert werden oder ist zu rigide, um eine Lösung „zu entwickeln“.
- Die Anschaffung eines Treasury-Management-Systems oder eines professionellen Bewertungsprogramms lohnt sich nicht, da dessen Lizenz- und Implementierungskosten nicht im Einklang mit der Häufigkeit und Komplexität der Aufgabenstellung stehen.

Die Lücke zwischen Tabellenkalkulation und Treasury-Management-System zu schließen, kann sich jedoch schwieriger gestalten, als man in Zeiten erwarten würde, in denen es im Internet für nahezu jede Aufgabe eine breite Auswahl an kostenlosen und kostenpflichtigen Programmen gibt. Viele Angebote unterhalb des professionellen Segments sind zu grob geschnitten und ermöglichen bestenfalls eine erste Annäherung an ein Thema.

Vor diesem Hintergrund ist „babz“ entstanden – die Bibliothek zum Aufbau und zur Bewertung von Zahlungsströmen. „babz“ ist eine Sammlung verschiedener Funktionen, die eine Bearbeitung finanzmathematischer Aufgaben unterhalb des Segments professioneller Anwendungen

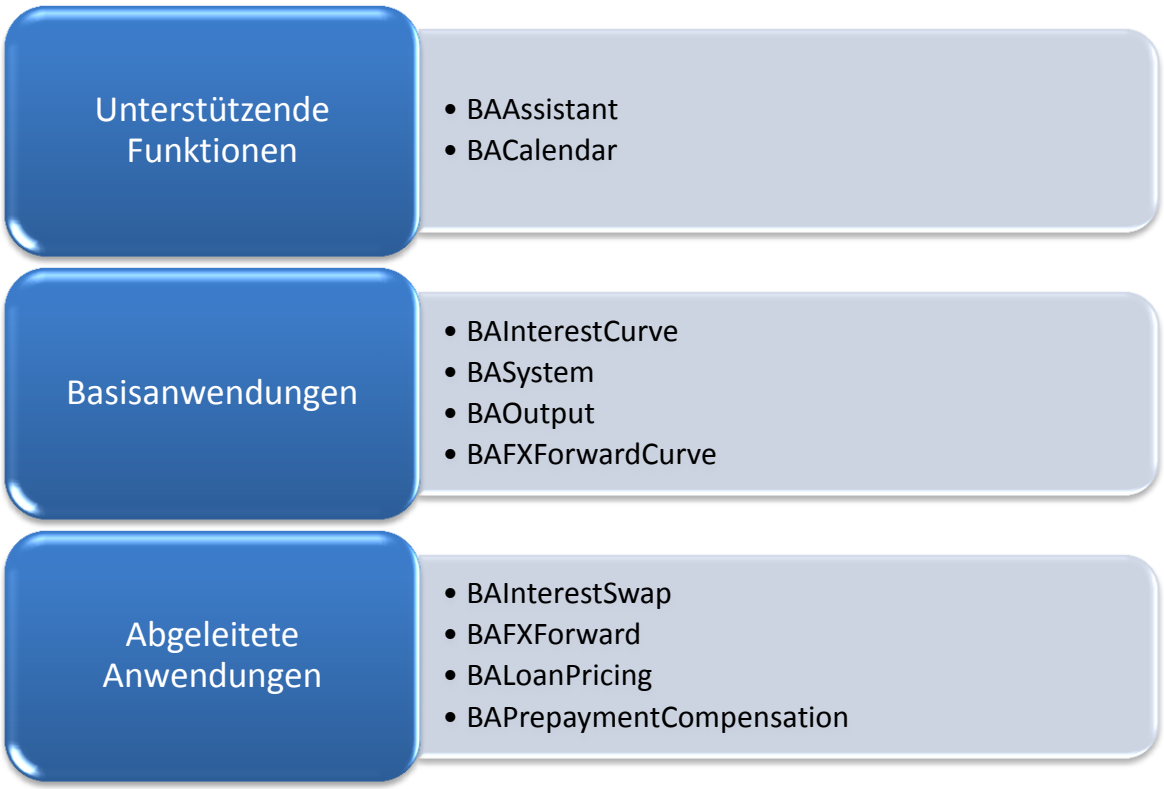
ermöglichen soll. Es handelt sich dabei nicht um ein Anwenderprogramm, das dem Nutzer eine Arbeitsumgebung mit verschiedenen Fenstern, Dialogen und Menüs anbietet, sondern um einen „Rechenkern“, auf dessen Funktionen von Anwendungsprogrammen zugegriffen werden kann.

Aus der vorgenannten Zielsetzung ergeben sich folgende Restriktionen, die bei der Anwendung des „babz“-Rechenkerns nebst sämtlichen Zubehörs (Dokumentationen, Schnittstellen, Erweiterungen) unbedingt zu beachten sind:

Die Software ist nicht für den professionellen Einsatz bestimmt. Mit der Software erstellte Berechnungsergebnisse dürfen nicht nur Bestimmung von Handelspreisen oder zur Begründung von Handelsentscheidungen herangezogen werden. Mit der Software ermittelte Berechnungsergebnisse dürfen nicht als Beleg eines Marktpreises, Auflösungspreises oder einer sonstigen Kennzahl, Bestands- oder Bewegungsgröße verwendet werden. Mit der Software erstellte Berechnungsergebnisse können, bei identischen Parametern und Marktdaten, von den Ergebnissen anderer Computerprogramme und von den Berechnungsergebnissen verschiedener Marktteilnehmer abweichen. Die der Software zugrunde gelegten Berechnungsvorschriften (mathematische Formeln, Algorithmen) können von den Berechnungsvorschriften anderer Computerprogramme, verschiedener Marktteilnehmer sowie von an den Märkten allgemein anerkannten Berechnungsvorschriften (Usancen) abweichen. Die in der Software programmierten Berechnungsvorschriften können von den in der Dokumentation beschriebenen Berechnungsvorschriften abweichen, soweit Änderungen an der Software vorgenommen wurden, die noch keinen Eingang in die Dokumentation gefunden haben. Im Rahmen der Entwicklung der Software können Fehler bei der Formulierung und Programmierung von Berechnungsvorschriften unterlaufen sein, wodurch Berechnungsergebnisse fehlerhaft sein könnten. Während des Betriebs der Software können Fehler auftreten, wodurch Berechnungsergebnisse verfälscht werden könnten (Laufzeitfehler).

1.2 Technik & Aufbau

Der Rechenkern ist in einer ActiveX®-DLL untergebracht, die mit Visual Basic® 6.0 (SP6) erstellt wurde. Folgende Klassen sind enthalten:



Die Klassen der Kategorie „Unterstützende Funktionen“ stellen den übrigen Klassen grundlegende Berechnungen und Daten zur Verfügung:

- **BAAssistant**

  Die Klasse *BAAssistant* bietet grundlegende Funktionen zur Unterstützung der anderen Klassen. In der aktuellen Version sind lediglich zwei Funktionen zur Interpolation von Daten sowie eine Funktion zur Rückgabe von Konstantenbezeichnungen implementiert.

- **BACalendar**

  Die Klasse ***BACalendar*** bietet Funktionalitäten an, um Kalender unter der Berücksichtigung

frei definierbarer Bankfeiertage zu programmieren, auf deren Basis Daten ermittelt und manipuliert sowie Zinstage, Day Count Fractions usw. berechnet werden.

- **BAInterestCurve**

  Objekte der Klasse *BAInterestCurve* speichern Zinskurven und bereiten diese für die Verwendung in anderen Klassen auf. Zentrale Aufgabe ist die Ableitung von Abzinsungsfaktoren (Diskontfaktoren) aus

- **BASystem**

  Mit der Klasse *BASystem* werden Parameter festgelegt, die sich auf grundsätzliche Einstellungen beziehen und das Verhalten mehrerer Klassen steuern. *BASystem* schaltet den Rechenkern mit einem geeigneten Registrierungsschlüssel frei und bietet Funktionen zum Debugging.

- **BAOutput**

  Objekte der Klasse *BAOutput* dienen der Aufnahme, Weiterverarbeitung und Auswertung von Daten. Beispielsweise schreiben Objekte der Klasse *BALoan* ihre Berechnungsergebnisse in ein *BAOutput-Objekt*, von wo aus diese z.B. für Barwertberechnungen herangezogen werden können.

- **BAFXForwardCurve**

  Mit Objekten der Klasse *BAFXForwardCurve* werden Kassa- und Termin-Wechselkurse in Form einer Kurve abgebildet und für die Verwendung in anderen Klassen vorbereitet. Die zentrale Aufgabe ist die Bereitstellung von Kursen zur Bewertung von Devisengeschäften.

Die Klassen der Kategorie „Basisanwendungen“ stellen Berechnungen und Funktionen bereit, die entweder für sich alleine stehen (z.B. ein Darlehen berechnen) oder aber die Grundlage für

weitergehende Anwendungen bilden (z.B. einen Zinsswap bewerten, der aus zwei Darlehen aufgebaut wird):

- **BALoan**

  Die Klasse *BALoan* berechnet anhand verschiedener Parameter Bestands- und Bewegungsgrößen von Darlehen. Die berechneten Daten werden als Klasse *BAOutput* zurückgegeben und können so für weitere Anwendungen herangezogen werden (z.B. Einstandsermittlung, Vorfälligkeitsentschädigung, Barwerte).

- **BAFXForward**

  Mit der Klasse *BAFXForward* können Devisentermingeschäfte (Solo-Termingeschäfte/Outrights) bewertet werden.

Die Kategorie „Abgeleitete Anwendungen“ fasst schließlich die Klassen zusammen, die aufbauend auf den Basisanwendungen weiterführend Berechnungsmöglichkeiten bieten:

- **BAInterestSwap**

  Die Klasse *BAInterestSwap* dient der Abbildung und Bewertung von Zinsswaps. Abgeleitet aus der Bewertung kann der faire Festsatz und der faire Spread ermittelt werden.

- **BAPrepaymentCompensation**

  Mit Objekten der Klasse *BAPrepaymentCompensation* kann für vorfällig abzulösende Darlehen eine Vorfälligkeitsentschädigung berechnet werden.

- **BALoanPricing**

  Die Klasse *BALoanPricing* dient der Ermittlung von fairen Darlehenszinssätzen auf Grundlage

einer gegebenen Zinskurve. So kann z.B. ein von einem Kreditinstitut angegebener Zinseinstand nachvollzogen werden.

1.3 Fehlerbeseitigung und Optimierung

Es wird sich nicht vermeiden lassen, dass Sie bei der Arbeit mit „babz“ Fehler finden werden, Funktionen, die andere Ergebnisse zu Tage fördern, als zu erwarten gewesen wäre und Strukturen, die man hätte geschickter gestalten können. Sie werden Rechtschreibfehler und möglicherweise auch zu aktualisierende oder unvollständige Darstellungen in diesem Handbuch finden. Schließlich werden Sie kleine und umfangreichere Funktionen vermissen, die sich auch nicht im nachfolgenden Abschnitt „“ finden. Tragen Sie bitte zur Verbesserung von „babz“ bei und senden Sie Ihre Hinweise an

entwicklung@babz-software.de

Fügen Sie Ihrer Nachricht bitte Hinweise bei, die für die Nachvollziehbarkeit des Fehlers hilfreich sind (Berechnungsparameter, Berechnunsergebnisse, mitgeschnittene Trace-Daten, das von Ihnen erwartete Ergebnis, Vergleichsergebnisse aus einer anderen Anwendung etc.).

1.4 Ausblick

Zwischen den ersten Vorläufern von babz und der aktuellen Version liegen rund sechs Jahre. In dieser Zeit sind zahlreiche neue Funktionen hinzugekommen, andere sind wieder verschwunden - erst in der Anwendung zeigt sich, welche Funktionen ihren Zweck wie gewünscht erfüllen und in welcher Richtung Ergänzungen und Änderungen erforderlich und sinnvoll sind. Das gilt auch für zukünftige Entwicklungen. Sofern weitere Versionen von babz entwickelt werden, ist als grober Pfad angedacht:

- Die Möglichkeit der Bewertung von bedingten Geschäften,
- die Implementierung verschiedener Kennzahlenberechnungen,
- ein Marktdatenspeicher, aus dem die verschiedenen Instrumente bedient werden können,
- der Ausbau der Funktionalitäten von Zinskurven,
- die Umsetzung im .NET-Framework[1].

1.5 Hinweise zum Handbuch

Dieses Handbuch beschreibt den babz-Rechenkern aus einer technischen Perspektive und ist als Nachschlagewerk für Programmierer gedacht, die auf den Rechenkern zugreifen. Jedoch kann die Dokumentation auch dem Endanwender eine Hilfe sein, wenn er die Funktionsweise einzelner Module verstehen möchte und die technischen Begriffe und Erläuterungen bei der Lektüre ausblendet.

In den nachfolgenden Kapiteln sind die einzelnen Klassen des Rechenkerns dokumentiert. Jede Klasse stellt eine logisch zusammengehörende Gruppe von Eigenschaften (Berechnungsparametern) und Methoden (Berechnungsanweisungen) dar. Entsprechend ist jedes Kapitel in die Hauptabschnitte „Aufgaben & Anwendung“ (kurze, allgemeine Beschreibung der Klasse), „Eigenschaften“, „Methoden“ und „Ereignisse“ gegliedert.

Den Ausführungen zu den einzelnen Eigenschaften und Methoden ist jeweils eine hellgraue Box vorangestellt, in der die Aufgabe sowie die wichtigsten Parameter und Merkmale der Eigenschaft bzw. Methode aufgeführt sind. Detaillierte Ausführungen sind, wo erforderlich, im Text unter der Box zu finden.

[1] babz wurde, wie oben beschrieben, mit Visual Basic® 6.0 erstellt, das bereits über 10 Jahre alt und somit nicht mehr zeitgemäß ist. Eine Umsetzung im aktuellen .NET-Framework ist grundsätzlich möglich, da babz bereits jetzt vollständig objektorientiert aufgebaut ist. Eine für .NET erstellte Version könnte jedoch nicht mehr unmittelbar in VBA-Projekt eingebunden werden (z.B. Verwendung in Microsoft Excel®) – daher die zögerliche Umstellung.

Technische Bezeichnungen, also Variablen, Konstanten, Klassennamen, Bezeichner etc. sind im Text kursiv dargestellt (Bsp.: *BALoan*). In Tabellen sind technische Bezeichnungen nicht gesondert hervorgehoben, wenn aus der Spaltenbezeichnung klar hervorgeht, dass es sich um technische Bezeichnungen handelt. Ebenso sind technische Bezeichnungen in den im vorherigen Abschnitt beschriebenen Boxen nicht gekennzeichnet.

Werte sind mit Anführungszeichen vom Text abgesetzt (Bsp.: '2'). Beispiele sind im Text eingerückt und kursiv dargestellt. Technische Bezeichnungen sind in Beispieltexten durch Anführungszeichen markiert.

Beispiel: *Mit 'Calculate' berechnen Sie den Fair Value des Zinsswaps.*

2 Klasse BASystem

2.1 Aufgaben & Anwendung

Mit der Klasse *BASystem* werden Parameter festgelegt, die sich auf grundsätzliche Einstellungen beziehen und das Verhalten mehrerer Klassen steuern. *BASystem* schaltet den Rechenkern mit einem geeigneten Registrierungsschlüssel frei und bietet Funktionen zum Debugging.

Grundsätzlich ist zu empfehlen, dass Sie vor der Erstellung von Objekten zur Durchführung von Berechnungen (z.B. *BALoan*, *BAInterestSwap* etc.) einen Verweis auf ein Objekt der Klasse *BASystem* erstellen, einen gültigen Registrierungsschlüssel laden, die erforderlichen Systemeinstellungen vornehmen und diesen Verweis erst nach Abschluss der Berechnungen entfernen. Die Systemeinstellungen bleiben erhalten, solange zumindest ein Verweis auf eine Klasse des Rechenkerns besteht – durch das Aufrechterhalten des Verweises auf *BASystem* ist dies gewährleistet.

2.2 Eigenschaften

2.2.1 AccruedInterestSettlementDate

Verwendung:	Zeitpunkt der Abgrenzung von Stückzinsen
Datentyp:	Boolean
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Für Berechnungen, bei denen Stückzinsen abzugrenzen sind, kann mit *AccruedInterestSettlementDate* eingestellt werden, ob die Abgrenzung auf das Fixing-Datum oder auf das Settlement-Datum der Zinskurve erfolgt, anhand derer die Berechnung durchgeführt wird (z.B. Berechnung der Stückzinsen eines Zinsswaps).

Wert	Hinweis
False (Standardwert)	Zinsabgrenzung auf das Fixing-Datum der Zinskurve
True	Zinsabgrenzung auf das Settlement-Datum der Zinskurve

Bitte beachten Sie die Hinweise zur Konsistenz der Berechnungsergebnisse im nachfolgenden Abschnitt „2.2.2 CashflowsStartDate“.

2.2.2 CashflowsStartDate

Verwendung:	Einbeziehung von Cashflows in die Barwertberechnung
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

CashflowsStartDate steuert, ab welchem Zeitpunkt Zahlungen in die Berechnung eines Barwertes einbezogen werden

Konstante	Wert	Hinweis
baCashflowsStartDateSettlementExcluding	0	Es werden alle Zahlungen einbezogen, die nach dem Settlement-Datum der Zinskurve liegen.
baCashflowsStartDateSettlementIncluding	1	Es werden alle Zahlungen einbezogen, die an oder nach dem Settlement-Datum der Zinskurve liegen.
baCashflowsStartDateFixingExcluding	2	Es werden alle Zahlungen einbezogen, die nach dem Fixing-Datum der Zinskurve liegen (Standard).
baCashflowsStartDateFixingIncluding	3	Es werden alle Zahlungen einbezogen, die an oder nach dem Fixing-Datum der Zinskurve liegen.

Die Berechnung von Barwerten erfolgt stets auf das Settlement-Datum einer Zinskurve. Zahlungen, die zwischen dem Fixing-Datum und dem Settlement-Datum liegen (jeweils einschließlich), werden mit dem Diskontfaktor 1,00 in den Barwert einbezogen.

Beachten Sie, dass die Einstellungen *CashflowsStartDate* und *AccruedInterestSettlementDate* miteinander in Zusammenhang stehen. Werden z.B. bei einem Zinsswap die Stückzinsen auf das Fixing-Datum berechnet, in die Barwertberechnung aber nur Zahlungsströme nach dem Settlement-Datum einbezogen, kann diese zu inkonsistenten Ergebnissen führen.

2.2.3 CurrRequired

Verwendung:	Gibt an, ob Objekte mit einer Währung versehen werden müssen
Datentyp:	Boolean
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Verschiedene Objekte können jeweils über die Eigenschaft *Curr* mit einem Währungscode versehen werden. Weisen zwei Objekte unterschiedliche Währungscodes auf, während eine Berechnung jedoch identische Währungscodes erfordert, tritt ein Laufzeitfehler auf. Kann auf einen Abgleich der Währungen verzichtet werden, müssen für die einzelnen Objekte keine Währungscodes festgelegt werden, wenn *CurrRequired* auf *False* gesetzt ist. Ist *CurrRequired* auf *True* gesetzt, muss auf jeden Fall ein Währungscode angegeben werden.

2.2.4 DigitsDiscountFactor

Verwendung:	Anzahl der Nachkommastellen für Diskontfaktoren
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Mit *DigitsDiscountFactor* wird gesteuert, auf wie viele Nachkommastellen Diskontfaktoren kaufmännisch gerundet werden. Erfolgt keine Angabe (Standardwert '0'), erfolgt keine Rundung.

2.2.5 DigitsInterestRate

Verwendung:	Anzahl der Nachkommastellen für Zinssätze
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Mit *DigitsInterestRate* wird gesteuert, auf wie viele Nachkommastellen Zinssätze kaufmännisch gerundet werden. Erfolgt keine Angabe (Standardwert '0'), erfolgt keine Rundung.

2.2.6 InterestCurveUserExit

Verwendung:	Verwendung eines externen Objekts zur Berechnung von Zinskurven
Datentyp:	Boolean
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Mit *InterestCurveUserExit* kann die Berechnung von Zinskurven und den daraus abgeleiteten Diskontfaktoren sowie Zinsterminsätzen an ein externes Objekt übergeben werden. Ist für *InterestCurveUserExit* der Wert *True* eingestellt, erfolgt die Berechnung durch das externe Objekt, andernfalls durch die internen Algorithmen der Klasse *BAInterestCurve*.

Wie das externe Objekt in die Berechnung einbezogen wird und wie das Objekt aufgebaut sein muss, ist in Abschnitt „4.2.23 UserExitObject“ auf Seite 52 dargestellt.

2.2.7 NoRegKeyToRegistry

Verwendung:	Deaktivierung des Zugriffs auf die Windows-Registry
Datentyp:	Boolean
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Wird mit *SetRegKey* ein gültiger Registrierungsschlüssel gesetzt, wird dieser in der Windows-Registry hinterlegt und bei einem erneuten Aufruf des Rechenkerns von dort geladen. Setzen Sie *NoAccessToRegistry* auf '*True*', um diesen Zugriff auf die Registry zu unterbinden.

2.2.8 NoRegKeyUserDialog

Verwendung:	Deaktivierung des Anwenderdialogs bei fehlendem Registrierungsschlüssel
Datentyp:	Boolean
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Kann eine Berechnung nicht durchgeführt werden, da kein gültiger Registrierungsschlüssel gesetzt wurde, wird zunächst geprüft, ob ein Registrierungsschlüssel aus der Windows-Registry geladen werden kann (sofern *NoAccessToRegistry* nicht auf *'True'* gesetzt wurde). Sofern kein Schlüssel aus der Registry geladen werden kann, wird ein Anwenderdialog eingeblendet, der zur Eingabe eines Schlüssels auffordert. Setzen Sie *NoRegKeyUserDialog* auf *'True'*, um diesen Dialog zu unterbinden.

2.2.9 RegKey

Verwendung:	Registrierungsschlüssel zur Freischaltung des Rechenkerns
Datentyp:	baRegKey
Lese-/Schreibzugriff:	Ja/Ja

Grundlegende Funktionen des Rechenkerns können erst nach Laden eines geeigneten Registrierungsschlüssels genutzt werden. Die Angabe des Registrierungsschlüssels erfolgt über eine Variable des Datentyps *baRegKey*:

Element	Datentyp	Hinweis
Name1	String	Name des Inhabers des Registrierungsschlüssels (1)
Name2	String	Name des Inhabers des Registrierungsschlüssels (2)
Expiration	Date	Datum, bis zu dem der Registrierungsschlüssel gültig ist.
Key	String	Registrierungsschlüssel

Nach Angabe des Registrierungsschlüssels muss dieser mit der Methode *SetRegKey* geladen werden.

2.2.10 Trace

Verwendung:	Protokollierung von Vorgängen
Datentyp:	Collection
Lese-/Schreibzugriff:	Ja/Ja
Mögliche Werte:	Werte vom Typ baTraceMessage

Wenn die Aufzeichnung von Vorgängen aktiviert ist (*TraceLevel*), werden die erfassten Informationen in der Aufzählung *Trace* in Form von Einträgen des Datentyps *baTraceMessage* gespeichert.

Element	Datentyp	Hinweis
Event	Byte	Art des Ereignisses
Time	Date	Datum & Uhrzeit
Level	Integer	Gliederungsebene
TraceClass	String	Klasse, in der das Ereignis aufgetreten ist
TraceSub	String	Prozedur, in der das Ereigniss aufgetreten ist
TraceMessage	String	Informationen zum Ereignis

ErrorNumber	Long	Fehlernummer, sofern ein Fehler aufgetreten ist

Event kann folgende Werte annehmen:

Konstante	Wert	Hinweis
baTraceEnterSub	1	Eintritt in eine Prozedur
baTraceExitSub	2	Verlassen einer Prozedur
baTraceError	4	Laufzeitfehler
baTraceStep	8	Zwischenschritt (innerhalb einer Prozedur)

Level erleichtert die Auswertung der in *Trace* gespeicherten Nachrichten, indem diese „eingerückt“ werden. Bei jedem Eintritt in eine Prozedur wird *Level* um einen Zähler erhöht, bei jedem Verlassen einer Prozedur um einen Zähler herabgesetzt.

2.2.11 TraceCache

Verwendung:	Anzahl der in Trace vorzuhaltenden Einträge
Datentyp:	Integer
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

TraceCache legt die Anzahl der Einträge fest, die maximal in *Trace* gespeichert werden. Würde die festgelegte Anzahl durch Hinzufügen eines neuen Eintrags überschritten, wird der älteste Eintrag gelöscht. Bei der Standardeinstellung '*0*' werden keine Einträge entfernt.

2.2.12 TraceLevel

Verwendung:	Festlegung der zu protokollierenden Ereignisse
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Mit *TraceLevel* wird festgelegt, ob und welche Ereignisse protokolliert werden sollen. Mit dem Standardwert '*0*' erfolgt keine Protokollierung. Durch Addition der im Kapitel „2.2.10 Trace“ (Seite 23) für *Event* aufgeführten Werte können die aufzuzeichnenden Ereignisse ausgewählt werden. Die Werte *baTraceEnterSub* und *baTraceExitSub* können nur gemeinsam verwendet werden. Wird eines dieser beiden Ereignisse ausgewählt, wird das jeweils andere automatisch in die Protokollierung aufgenommen. Addieren Sie zusätzlich den Wert *baTracePerseverative*, um häufig wiederholte Prozeduren (z.B. *GetEasterSunday*, *IsBankHoliday*) in die Protokollierung aufzunehmen. Diese werden standardmäßig nicht aufgezeichnet, um die Lesbarkeit des Protokolls durch die Fokussierung auf relevante Punkte zu verbessern.

Beachten Sie bitte, dass die Protokollierung von Ereignissen zu Lasten der Performance geht. Es wird empfohlen, die Protokollierung nur zur Analyse von Fehlern einzuschalten.

2.3 Methoden

2.3.1 ClearTrace

Funktion:	Löscht alle in Trace enthaltenen Einträge
Parameter:	Keiner
Rückgabewert:	Keiner

2.3.2 LoadRegKey

Funktion:	Laden des Registrierungsschlüssels aus der Windows-Registry
Parameter:	Keiner
Rückgabewert:	Keiner

LoadRegKey lädt einen in der Windows-Registry gespeicherten Registrierungsschlüssel. Anschließend kann der Rechenkern mit *SetRegKey* freigeschaltet werden. *LoadRegKey* greift auch dann auf die Registry zu, wenn *NoAccessToRegistry* auf '*True*' gesetzt wurde.

2.3.3 SetRegKey

Funktion:	Freischalten des Rechenkerns
Parameter:	Keiner
Rückgabewert:	Keiner

SetRegKey schaltet den Rechenkern mit dem in *RegKey* hinterlegten Registrierungsschlüssel frei. Ist der Schlüssel nicht gültig, tritt ein Fehler auf.

Sofern *NoAccessToRegistry* nicht auf '*True*' gesetzt ist, wird der (gültige) Registrierungsschlüssel in der Windows-Registry unter '*HKEY_Current_User\babz*' abgespeichert. Er kann von dort mit *LoadRegKey* geladen werden. Er wird automatisch geladen, wenn noch kein gültiger Registrierungsschlüssel gesetzt wurde, für die weiteren Berechnungsschritte jedoch erforderlich ist und *NoAccessRegistry* nicht auf '*True*' gesetzt wurde.

2.3.4 WriteTraceToFile

Funktion:	Speichern der in Trace enthaltenen Einträge in einer Textdatei
Parameter:	Filename (String)
Rückgabewert:	Keiner

Nach erfolgreichem Aufruf enthält die angegebene Datei die in *Trace* enthaltenen Einträge. Pro Zeile wird ein Eintrag mit Tab-getrennten Daten gespeichert.

3 Klasse BACalendar

3.1 Aufgaben & Anwendung

Die Klasse ***BACalendar*** bietet Funktionalitäten an, um Kalender unter der Berücksichtigung frei definierbarer Bankfeiertage zu programmieren, auf deren Basis Daten ermittelt und manipuliert sowie Zinstage, Day Count Fractions usw. berechnet werden.

3.2 Eigenschaften

3.2.1 Description

Verwendung:	Erklärende Bezeichnung des Objekts
Datentyp:	String
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein
Mögliche Werte:	Keine Beschränkung

3.2.2 Calendars

Verwendung:	Definition der zu ladenden vordefinierten Kalender
Datentyp:	Collection
Lese-/Schreibzugriff:	Ja/Nein

Calendars legt fest, welche Kalender mit der *Load-Methode* geladen werden. Die nachfolgend aufgeführten Konstanten können der Auflistung hinzugefügt werden, der Aufruf von *Load* fügt der *Holidays-Auflistung* die jeweils mit dem Kalender verknüpften Feiertage hinzu.

Konstante	Wert	Hinweis
baNone	0	Kein Kalender geladen
baCalendarTarget	1	Target-Feiertage
baCalendarGermany	2	Bundeseinheitliche Feiertage (Deutschland)

baCalendarNYSEEuronext	3	Feiertage NYSE Euronext
baCalendarUSA	4	Gesetzliche Feiertage USA
baCalendarUK	5	Gesetzliche Feiertage Großbritannien (England)
baCalendarCZ	6	Feiertage Tschechische Republik

Eine Auflistung, welche Tage in den jeweiligen Kalendern enthalten sind, finden Sie im Abschnitt „3.3.9 Load“ auf Seite 38.

Wurde ein Kalender nicht durch Aufruf eines vordefinierten Kalenders mit der *Load-Methode* geladen, sondern durch Hinzufügen einzelner Feiertage zur *Holidays-Auflistung* aufgebaut, gibt *Calendars* stets eine leere Auflistung zurück. Es wird nicht überprüft, ob die einzeln hinzugefügten Feiertage insgesamt einem vordefinierten Kalender entsprechen.

3.2.3 Holidays

Verwendung:	Definition der Bankfeiertage
Datentyp:	Collection
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein
Mögliche Werte:	Werte des Typs baHoliday

Um für ein Objekt der Klasse *BACalendar* Bankfeiertage zu definieren, werden diese der *Holidays-Auflistung* hinzugefügt. Es bestehen folgende Möglichkeiten, einen Bankfeiertag mittels einer Variable des Typs *baHoliday* zu definieren:

Vordefinierte Feiertage

Für das Element *Predefined* können folgende Konstanten gesetzt werden:

Konstante	**Wert**	**Hinweis**
baSunday	1	Sonntag
baMonday	2	Montag

baTuesday	3	Dienstag	
baWednesday	4	Mittwoch	
baThursday	5	Donnerstag	
baFriday	6	Freitag	
baSaturday	7	Samstag	
baNewYearsDay	8	Neujahr (1. Januar)	
baNewYearsEve	9	Silvester (31. Dezember)	
baChristmasEve	10	Heiligabend (24. Dezmeber)	
baChristmasDay	11	1. Weihnachtsfeiertag (25. Dezember)	
baBoxingDay	12	2. Weihnachtsfeiertag (26. Dezember)	
baGoodFriday	13	Karfreitag (Ostersonntag – 2 Tage)	
baEasterSunday	14	Ostersonntag	
baEasterMonday	15	Ostermontag (Ostermontag + 1 Tag)	
baWhitMonday	16	Pfingstmontag (Ostermontag + 50 Tage)	
baAscensionDay	17	Christi Himmelfahrt (Ostermontag + 39 Tage)	
baGermanUnificationDay	18	Tag der Deutschen Einheit (3. Oktober)	
baLabourDay	19	Tag der Arbeit (1. Mai)	
baMartinLutherKingDay	20	Martin Luther King Day (Dritter Montag im Januar)	USA
baWashingtonsBirthday	21	Washingtons Birthday (Dritter Montag im Februar)	USA
baMemorialDay	22	Memorial Day (Letzter Montag im Mai)	USA
baIndependenceDay[1)]	23	Independence Day (4. Juli)	USA
baLaborDayUSA	24	Labor Day (Erster Montag im September)	USA
baColumbusDay	25	Columbus Day (Zweiter Montag im Oktober)	USA
baVeteransDay	26	Veterans Day (11. November)	USA
baThanksgivingDay	27	Thanksgiving Day (Vierter Donnerstag im November)	USA
baChristmasDayUSA[1)]	28	Christmas Day (25. Dezember)	USA
baNewYearsDayUSA[1)]	29	New Years Day (1. Januar)	USA
baEarlyMayBankHoliday	30	Early May Bank Holiday (Erster Montag im Mai)	UK
baSpringBankHoliday	31	Spring Bank Holiday (Letzter Montag im Mai)	UK
baSummerBankHoliday	32	Summer Bank Holiday (Letzter Montag im August)	UK
baNewYearsDayUK[2)]	33	New Years Day (1. Januar)	UK
baChristmasDayUK[2)]	34	Christmas Day (25. Dezember)	UK

baBoxingDayUK[2)]	35	Boxing Day (26. Dezember)	UK
baVictoryDayCZ	36	Tag des Sieges (8. Mai)	CZ
baSaintCyrilandMethodi-usDay	37	Tag der Slawenapostel Kyrill und Method (5. Juli)	CZ
baJanHusDay	38	Tag der Verbrennung von Jan Hus (6. Juli)	CZ
baDayOfTheCzechState	39	Tag des tschechischen Staates (28. September)	CZ
baDayOfTheCzechoslovak-Republic	40	Tag der Entstehung eines selbstständigen tschechoslowakischen Staates (28. Oktober)	CZ
baDayOfFreedomAndDemocracyCZ	41	Tag des Kampfes für Freiheit und Demokratie (17. November)	CZ

Fallen die mit [1)] markierten Feiertage auf einen Sonntag, so wird der folgende Montag ebenfalls als Feiertag interpretiert, fallen sie auf einen Samstag, ist der vorhergehende Freitag ebenfalls ein Feiertag.

Fallen die mit [2)] markierten Feiertage auf ein Wochenende, wird der folgende Montag als Feiertag interpretiert. Ausnahme: Der 2. Weihnachtsfeiertag in Großbritannien (*baBoxingDayUK*) wird auf den folgenden Dienstag geschoben, wenn der 1. Weihnachtsfeiertag (*baChristmasDayUK*) auf den Montag verschoben wird.

Sie können die *AddPredefinedHoliday*-Methode verwenden, um vordefinierte Feiertage hinzuzufügen. Dies ist komfortabler als Variablen des Typs *baHoliday* zu füllen und der *Holidays-Auflistung* hinzuzufügen.

Variable Feiertage (abhängig vom Ostersonntag)

Variable Feiertage fallen nicht zwingend in jedem Kalenderjahr auf das gleiche Datum, sondern sind in der Regel vom Ostersonntag abhängig. Für das Element *EasterSunday* kann daher angegeben werden, wie viele Tage vor oder nach dem Ostersonntag eines Jahres ein Bankfeiertag berücksichtigt werden soll.

Beispiel: *Wird das Element 'EasterSunday' einer Variable vom Typ 'baHoliday' auf den Wert '1' gesetzt und die Variable der 'Holidays'-Auflistung hinzugefügt, wird der Ostermontag als Feiertag berücksichtigt. Der Wert '-2' setzt den Karfreitag als Feiertag ein.*

Variable Feiertage (bestimmter Wochentag in einem Monat)

Variable Feiertage können auch immer auf einen bestimmten Wochentag eines Kalendermonats fallen (z.B. wird in den USA Thanksgiving immer am 4. Donnerstag im November gefeiert). Solche Feiertage können mit den Elementen *Month*, *WeekDay* und *WeekDayCount* definiert werden. *WeekDay* bezeichnet den Wochentag, wobei die Konstanten *baSunday* bis *baSaturday* genutzt werden können (siehe Tabelle oben). *WeekDayCount* gibt an, auf den wievielten Wochentag des Monats *Month* der Feiertag fällt. Wird *WeekDayCount* mit negativem Vorzeichen angegeben, erfolgt die Zählung vom Ende des Monats her (letzter Wochentag, vorletzter Wochentag, ...).

Beispiel: *Wird das Element 'Month' einer Variable vom Typ 'baHoliday' auf den Wert '9' gesetzt, 'Weekday' auf 'baMonday', 'WeekdayCount' auf '1' und die Variable der 'Holidays'-Auflistung hinzugefügt, wird der erste Montag im September als Feiertag berücksichtigt (Tag der Arbeit in den USA).*

Feste Feiertage

Feste Feiertage Fallen in jedem Kalenderjahr auf das gleiche Datum, z.B. Silvester oder Neujahr. Um feste Feiertage zu definieren, setzen Sie das Element *Day* auf den gewünschten Tag, das Element *Month* auf den gewünschten Monat.

Ist ein Wert für das Element *Predefined* gesetzt, werden evtl. in den Elementen für variable und feste Feiertage gespeicherte Werte ignoriert. Ist ein Wert für das Element *EasterSunday* gesetzt, werden evtl. in den Elementen *Day, Month, WeekDay und WeekDayCount* (feste Feiertage und variable Feiertag, die vom Wochentag abhängig sind) gespeicherte Werte ignoriert. Zuletzt haben *Month*, *WeekDay* und *WeekDayCount* Vorrang vor festen Feiertagen.

Ausnahmeregeln

Für vom Ostersonntag abhängige variable Feiertage, auf bestimmte Wochentage fallende Feiertage sowie feste Feiertage können Ausnahmeregeln hinterlegt werden. Fügen Sie dazu der *Exeptions*-Auflistung eine oder mehrere Variablen des Typs *baHolidayException* zu.

Element	Datentyp	Hinweis
ExceptionalYear	Integer	Kalenderjahr, für das die Ausnahmeregel gelten soll
OnlyExceptionalYear	Boolean	Der Feiertag soll nur im in der Ausnahmeregel definierten Kalenderjahr gelten.
NotExceptionalYear	Boolean	Der Feiertag soll nicht im in der Ausnahmeregel definierten Kalenderjahr gelten.
ShiftHoliday	Boolean	Der Feiertag soll im in der Ausnahmeregel definierten Kalenderjahr auf ein anderes Datum verlegt werden.
ExceptionalDate	Date	Datum, auf das der Feiertag bei Anwendung von ‚ShiftHoliday' verlegt wird.

Die Feiertagsdefinition beschreibt die Regel, nach welcher der Feiertag in jedem Kalenderjahr ermittelt wird. Von dieser Regel wird abgewichen, wenn eine oder mehrere Ausnahmeregeln zum Feiertag hinterlegt sind:

- Enthält mindestens eine Ausnahmeregel für *OnlyExceptionalYear* den Wert *True*, wird der Feiertag für alle Kalenderjahre, für die keine Ausnahmeregel hinterlegt wurde, ignoriert. Für alle Kalenderjahre, für die eine Ausnahmeregel hinterlegt wurde, wird der Feiertag nach der in der Ausnahmeregel definierten Logik ermittelt.
- Enthält die Ausnahmeregel für *NotExceptionalYear* den Wert *True*, wird der Feiertag für das festgelegte Kalenderjahr ignoriert.
- Wird der Feiertag nach den beschriebenen Regeln für *OnlyExceptionalYear* und *NotExceptionalYear* berechnet (d.h. er ist für das Kalenderjahr explizit vorgesehen oder wurde für das Kalenderjahr nicht gestrichen), kann der auf ein festes, in *ExceptionalDate* zu hinterlegendes Datum verschoben werden, indem *ShiftHoliday* auf *True* gesetzt wird.

Andernfalls wird der Feiertag auf dem anhand der Feiertagsdefinition berechneten Datum belassen.

Für vordefinierte Feiertage können keine Ausnahmeregeln hinterlegt werden.

3.2.4 Tag

Verwendung:	Frei verwendbares Merkmal
Datentyp:	Variant
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein
Mögliche Werte:	Keine Beschränkung

3.3 Methoden

3.3.1 AddDateInverval

Funktion:	Fügt einem Datum ein Zeitintervall hinzu
Parameter:	BaseDate (Date)
	Interval (Byte)
	IntervalCount (Integer), optional
Rückgabewert:	Berechnetes Datum (Date)

Dem angegebenen Datum (*BaseDate*) wird ein festgelegtes Zeitintervall hinzugerechnet. Wird für *IntervalCount* ein Wert >1 angegeben, wird das Intervall entsprechend mehrfach addiert. Zurückgegeben wird das Datum, an dem die hinzugefügte Zeitspanne endet.

Konstante	Wert	Zeitintervall
baDay	1	1 Tag
baWeek	2	1 Kalenderwoche
baMonth	3	1 Kalendermonat

baQuarter	4	3 Kalendermonate
baYearSemi	5	6 Kalendermonate
baYear	6	12 Kalendermonate

3.3.2 AddPredefinedHoliday

Funktion:	Fügt dem Kalender vordefinierte Bankfeiertage hinzu
Parameter:	Holiday (Byte)
Rückgabewert:	Keiner

Anstatt eine Variable vom Typ *baHoliday* zu befüllen und der *Holidays-Auflistung* hinzuzufügen, können Sie einen vordefinierten Bankfeiertag mit der Methode *AddPredefinedHoliday* hinzufügen.

Beispiel: *Die Anweisung 'Calendar.AddPredefinedHoliday baNewYearsDay' fügt einem Objekt 'Calendar' der Klasse 'BACalendar' den Feiertag Neujahr hinzu.*

3.3.3 Clear

Funktion:	Entfernen aller Feiertage aus der Holidays-Auflistung
Parameter:	Keine
Rückgabewert:	Keiner

3.3.4 GetDateFromBusinessDays

Funktion:	Addition von Arbeitstagen zu einem Datum
Parameter:	BaseDate (Date)
	Businessdays (Integer)
	Invert (Boolean, optional)
Rückgabewert:	Berechnetes Datum (Date)

Berechnet ein Datum, indem zum Datum *BaseDate* die mit *Businessdays* vorgegebene Anzahl an Arbeitstagen hinzugezählt wird. Wird *Invert* auf *True* gesetzt, erfolgt die Berechnung rückwärts (die Arbeitstage werden vom Datum *BaseDate* abgezogen).

3.3.5 GetDayCountFraction

Funktion:	Berechnung der Day Count Fraction zwischen zwei Terminen
Parameter:	DayCountConvention (Byte)
	StartDate (Date)
	EndDate (Date)
Rückgabewert:	Day Count Fraction (Single)

Ermittelt den Teiler aus Zinstagen zwischen den beiden Daten und den Zinstagen eines Kalenderjahres (Day Count Fraction). Für die Berechnung der Zinstage wird, abhängig von der angegebenen Day Count Convention wie folgt verfahren:

Konstante	Wert	Hinweis
ba30360	1	30/360 Jeder Monat wird mit 30 Tagen gezählt, jedes Jahr mit 360 Tagen
ba30365	2	30/365 Jeder Monat wird mit 30 Tagen gezählt, jedes Jahr mit 365 Tagen
baAct360	4	Act/360 Jeder Monat wird mit der tatsächlichen Anzahl an Tagen gezählt, jedes Jahr mit 360 Tagen
baAct365	8	Act/365 Jeder Monat wird mit der tatsächlichen Anzahl an Tagen gezählt, jedes Jahr mit 365 Tagen
baActAct	16	Act/Act Monate und Jahre werden mit der jeweils tatsächlichen Anzahl an Tagen gezählt

3.3.6 GetInterestDays

Funktion:	Liefert die Anzahl der Zinstage zwischen zwei Daten zurück
Parameter:	DayCountConvention (Byte)
	StartDate (Date)
	EndDate (Date)
Rückgabewert:	Anzahl der Zinstage (Integer)

Die Zinstage werden gemäß der angegebenen Day Count Convention gezählt. Details zur Day Count Convention finden Sie im Kapitel „3.3.5 GetDayCountFraction“ (Seite 36).

3.3.7 GetYearInterestDays

Funktion:	Liefert die Anzahl der Zinstage eines Kalenderjahres
Parameter:	DayCountConvention (Byte)
	StartDate (Date)
	EndDate (Date)
Rückgabewert:	Anzahl der Zinstage (Integer)

Die Zinstage wird gemäß der angegebenen Day Count Convention ermittelt. Details zur Day Count Convention finden Sie im Kapitel „3.3.5 GetDayCountFraction“ (Seite 36).

3.3.8 IsBankHoliday

Funktion:	Prüft, ob es sich bei einem Datum um einen Feiertag handelt
Parameter:	CheckDate (Date)
Rückgabewert:	Ergebnis der Prüfung (Integer)

Fällt das angegebene Datum (*CheckDate*) auf einen Feiertag, gibt *IsBankHoliday* den Index (beginnend mit '1') desjenigen Elements in der *Holidays-Auflistung* zurück, das den Feiertag ausgelöst hat. Fällt das Datum nicht auf einen Feiertag, wird der Wert '0' zurückgegeben.

3.3.9 Load

Funktion:	Lädt einen vordefinierten Kalender
Parameter:	NoClear (Boolean, optional)
Rückgabewert:	Keiner

„Ab Werk“ beherrscht babz die nachfolgenden vordefinierten Kalender, die mittels Hinzufügen zur *Calendars-Auflistung* und Aufruf der *Load*-Methode geladen werden können:

Konstante	Wert	Hinweis
baCalendarTarget	1	Target Samstage, Sonntage, Neujahr, Karfreitag, Ostermontag, Tag der Arbeit, 1. Weihnachtsfeiertag, 2. Weihnachtsfeiertag
baCalendarGermany	2	Bundeseinheitliche Feiertage (Deutschland) Samstage, Sonntage, Neujahr, Karfreitag, Ostermontag, Tag der Arbeit, 1. Weihnachtsfeiertag, 2. Weihnachtsfeiertag, Pfingstmontag, Christi Himmelfahrt, Tag der Deutschen Einheit
baCalendarNYSEEuronext	3	NYSE Euronext Samstage, Sonntage, Neujahr (USA), Martin Luther King Day, Washingtons Birthday, Karfreitag, Memorial Day, Independence Day, Labor Day (USA), Thanksgiving Day, Christmas Day (USA)
baCalendarUSA	4	Gesetzliche Feiertage USA Samstage, Sonntage, Neujahr (USA), Martin Luther King Day, Washingtons Birthday, Memorial Day, Independence Day, Labor Day (USA), Columbus Day, Veterans Day, Thanksgiving Day, Christmas Day (USA)
baCalendarUK	5	Gesetzliche Feiertage Grobritannien (England) Samstage, Sonntage, Neujahr (UK), Karfreitag, Ostermontag, Early May Bank Holiday, Spring Bank Holiday, Summer Bank Holiday, Christmas Day (UK), Boxing Day (UK)
baCalendarCZ	6	Feiertage Tschechische Republik

	Samstage, Sonntage, Neujahr, Ostermontag, Tag der Arbeit, Tag des Sieges, Tag der Slawenapostel Kyrill und Method, Jan Hus Tag, Tag des tchechischen Staates, Tag der Entstehung eines selbständigen tschechoslowakischen Staates, Tag des Kampfes für Freiheit und Demokratie, Heiligabend, 1. Weihnachtsfeiertag, 2. Weihnachtsfeiertag

Samstage sind zwar keine Feiertage, meist kann samstags jedoch nicht gehandelt werden, weshalb der Samstag in allen vordefinierten Kalendern enthalten ist.

Eventuell bereits in der *Holidays-Auflistung* enthaltene Elemente werden gelöscht, sofern *NoClear* nicht auf *True* gesetzt wird.

3.3.10 SetDay

Funktion:	Legt ein Datum innerhalb eines Monats fest
Parameter:	UnadjustedDate (Date)
	Day (Byte)
Rückgabewert:	Angepasstes Datum (Date)

Der Tag des angegebenen Datums (*UnadjustedDate*) wird ignoriert und durch den angegebenen Tag (*Day*) ersetzt. Dabei wird der angegebene Monat nicht verlassen, so dass stets maximal der letzte Tag des Monats zurückgegeben wird.

Beispiel: *Mit UnadjustedDate '01.02.2007' und Day '31' liefert 'SetDay' das Datum '28.02.2007' zurück.*

3.3.11 ShiftDate

Funktion:	Verschiebt ein Datum, falls es auf einen Bankfeiertag fällt
Parameter:	UnadjustedDate (Date)
	BusinessDayConvention (Byte)
Rückgabewert:	Angepasstes Datum (Date)

Fällt das angegebene Datum (*UnadjustedDate*) auf einen Bankfeiertag, wird es abhängig von der *BusinessDayConvention* wie folgt behandelt:

Konstante	Wert	Behandlung des Datums
baNone	0	Keine Verschiebung
baPreceding	1	Verschiebung auf den ersten vorhergehenden Arbeitstag
baFollowing	2	Verschiebung auf den ersten folgenden Arbeitstag
baModifiedFollowing	4	Verschiebung auf den ersten folgenden Arbeitstag (Following), wenn dieser im selben Monat liegt wie das ursprüngliche Datum, ansonsten Verschiebung auf den ersten vorhergehenden Arbeitstag (Preceding)
baModifiedPreceding	8	Verschiebung auf den ersten vorhergehenden Arbeitstag (Preceding), wenn dieser im selben Monat liegt wie das ursprüngliche Datum, ansonsten Verschiebung auf den ersten nachfolgenden Arbeitstag (Following).
baAdjusted	128	Kann den o.g. Werten hinzugefügt werden, um bei Zins- und Tilgungszahlungen den Zeitpunkt der Zins- oder Tilgungsanrechnung zu definieren. Details im Kapitel „7.3.9 Interest“ auf Seite 87.

4 Klasse BAInterestCurve

4.1 Aufgaben & Anwendung

Objekte der Klasse *BAInterestCurve* speichern Zinskurven und bereiten diese für die Verwendung in anderen Klassen auf. Zentrale Aufgabe ist die Ableitung von Abzinsungsfaktoren (Diskontfaktoren) aus Geldmarkt- und Swapsätzen sowie aus Zero-Sätzen.

4.2 Eigenschaften

4.2.1 BusinessDayConvention_CashRates

Verwendung:	Business Day Convention der Geldmarktsätze
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

BusinessDayConvention_CashRates gibt an, mit welcher Business Day Convention diejenigen Zinssätze zu behandeln sind, die als Geldmarktsätze gekennzeichnet sind. Die Business Day Convention wird benötigt, um ausgehend vom Settlement-Date das Datum zu bestimmen, an dem die Periode endet, für den der Zinssatz jeweils gültig ist.

Details zu den Business Day Conventions finden Sie im Kapitel „3.3.11 ShiftDate“ auf Seite 40.

4.2.2 BusinessDayConvention_SwapRates

Verwendung:	Business Day Convention der Swap-Sätze
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

BusinessDayConvention_SwapRates gibt an, mit welcher Business Day Convention diejenigen Zinssätze zu behandeln sind, die als Swap-Sätze gekennzeichnet sind. Die Business Day Convention wird benötigt, um ausgehend vom Settlement-Date das Datum zu bestimmen, an dem die Periode endet, für den der Zinssatz jeweils gültig ist.

Details zu den Business Day Conventions finden Sie im Kapitel „3.3.11 ShiftDate“ auf Seite 40.

4.2.3 BusinessDayConvention_ZeroRates

Verwendung:	Business Day Convention der Zero-Sätze
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

BusinessDayConvention_ZeroRates gibt an, mit welcher Business Day Convention diejenigen Sätze zu behandeln sind, die als Zero-Sätze gekennzeichnet sind. Die Business Day Convention wird benötigt, um ausgehend vom Settlement-Date das Datum zu bestimmen, an dem die Periode endet, für den der Satz jeweils gültig ist.

Details zu den Business Day Conventions finden Sie im Kapitel „3.3.11 ShiftDate“ auf Seite 40.

4.2.4 Calendar

Verwendung:	Definition des Bankkalenders
Datentyp:	BACalendar
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Auf *Calendar* setzen alle Terminberechnungen (Ausrollen des Zeitgerüstes, Zinstage etc.) der Klasse auf. Ein *BACalendar*-Objekt muss auch dann zugewiesen werden, wenn keine Feiertage definiert werden sollen.

4.2.5 Curr

Verwendung:	Währung, für die die Zinskurve gültig ist
Datentyp:	String
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Abhängig von Systemeinstellung CurrRequired

Curr bestimmt, für welche Währung die Zinskurve gültig ist. Es können nur solche Objekte (z.B. Darlehen) die Zinskurve verwenden, welche dieselbe Währung aufweisen. Ist keine Währung angegeben, findet keine Kontrolle statt.

Ist die Systemeinstellung *CurrRequired* explizit auf *True* gesetzt, muss eine Währung angegeben werden. Nähere Informationen finden Sie im Abschnitt „2.2.3 CurrRequired“ auf Seite 20.

4.2.6 Description

Verwendung:	Erklärende Bezeichnung des Objekts
Datentyp:	String
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

4.2.7 DayCountConvention_CashRates

Verwendung:	Day Count Convention der Geldmarktsätze
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

DayCountConvention_CashRates gibt an, mit welcher Day Count Convention diejenigen Zinssätze im Rahmen der Berechnung der Diskontfaktoren zu behandeln sind, die als Geldmarktsätze gekennzeichnet sind.

Details zu den Day Count Conventions finden Sie im Kapitel „3.3.5 GetDayCountFraction“ auf Seite 36.

4.2.8 DayCountConvention_SwapRates

Verwendung:	Day Count Convention der Swap-Sätze
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

DayCountConvention_SwapRates gibt an, mit welcher Day Count Convention diejenigen Zinssätze im Rahmen der Berechnung der Diskontfaktoren zu behandeln sind, die als Swap-Sätze gekennzeichnet sind.

Details zu den Day Count Conventions finden Sie im Kapitel „3.3.5 GetDayCountFraction“ auf Seite 36.

4.2.9 DayCountConvention_ZeroRates

Verwendung:	Day Count Convention der Zero-Sätze
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

DayCountConvention_ZeroRates gibt an, mit welcher Day Count Convention diejenigen Sätze im Rahmen der Berechnung der Diskontfaktoren zu behandeln sind, die als Zero-Sätze gekennzeichnet sind.

Details zu den Day Count Conventions finden Sie im Kapitel „3.3.5 GetDayCountFraction“ auf Seite 36.

4.2.10 DiscountFactors

Verwendung:	Aus der Zinskurve abgeleitete Diskontfaktoren
Datentyp:	Collection
Lese-/Schreibzugriff:	Ja/Nein
Eingabe erforderlich:	Ja
Mögliche Werte:	Elemente des Typs baDiscountFactor

In der *DiscountFactors*-Auflistung sind nach dem erfolgreichen Aufruf der *Calculate*-Methode alle aus der Zinskurve abgeleiteten Diskontfaktoren und Zero-Sätze enthalten. Diese werden als Variablen des Typs *baDiscountFactor* gespeichert, der drei Elemente enthält:

Element	Datentyp	Hinweis
Date	Date	Datum, auf das sich der Diskontfaktor bezieht
DiscountFactor	Single	Diskontfaktor
ZeroRate	Single	Zero-Satz

DiscountFactor ist derjenige Faktor, mit dem eine Zahlung, die am Datum *Date* geleistet wird, multipliziert werden muss, um den Barwert dieser Zahlung am Settlement-Date der Zinskurve zu erhalten.

Zum Abfragen von Diskontfaktoren steht die *GetDF*-Funktion zur Verfügung, für Zero-Sätze die *GetZeroRate*-Funktion.

4.2.11 InterestRates

Verwendung:	Zinssätze der Zinskurve
Datentyp:	Collection
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja
Mögliche Werte:	Werte des Typs baInterestRate

Die *InterestRates*-Auflistung beschreibt alle Zinssätze der Zinskurve. Aus den Zinssätzen werden die Diskontfaktoren gewonnen. Die einzelnen Zinssätze werden durch Variablen des Typs *baInterestRate* beschrieben. Dieser Datentyp verfügt über die folgenden Elemente:

Element	Datentyp	Hinweis
Tenor	Single	Laufzeit
TenorUnit	Byte	Einheit der Laufzeit
InterestRate	Single	Zinssatz (excl. Spread)
InterestRateSpread	Single	Zinssatz (incl. Spread)
Date	Date	Datum (an dem der mit Tenor und TenorUnit beschriebene Zeitraum endet)
Type	Byte	Art des Zinssatzes
Source	String	Herkunft/Symbol des Zinssatzes

Tenor & TenorUnit

Mit *Tenor* und *TenorUnit* wird definiert, auf welche Laufzeit sich der Zinssatz bezieht, d.h. wann die Periode endet, für die der Zinssatz *InterestRate* gilt. Die Zeiteinheit wird mit *TenorUnit* angegeben, die Anzahl der Zeiteinheiten mit *Tenor*. Folgende Zeiteinheiten stehen zur Verfügung:

Konstante	Wert	Zeitintervall
baDay	1	1 Tag
baWeek	2	1 Kalenderwoche
baMonth	3	1 Kalendermonat

baQuarter	4	3 Kalendermonate
baYearSemi	5	6 Kalendermonate
baYear	6	12 Kalendermonate

Date

In *Date* ist nach erfolgreichem Aufruf der *Calculate*-Methode das Datum enthalten, an die mittels *Tenor* und *TenorUnit* definierte Zinsperiode endet (ausgehend von *Settlement*, unter Berücksichtigung der gültigen Business Day Convention).

InterestRate

InterestRate gibt den Nominalzinssatz p.a. oder den Zerosatz p.a. an. Der Wert muss in der Form „P/100“ vorliegen (4% = 4/100 = 0.04).

Type

Mit *Type* wird gekennzeichnet, ob es sich bei dem Satz um einen Geldmarktsatz, einen Swapsatz oder um einen Zerosatz handelt. Außerdem werden mit Type solche Sätze gekennzeichnet, die im Zuge des Bootstrapping interpoliert wurden:

Konstante	**Wert**	**Bedeutung**
baNone	0	Geldmarktsatz
baInterestRateInterpolated	1	Interpolierter Zinssatz
baInterestRateSwap	2	Swapsatz
baInterestRateZero	4	Zerosatz

Beispiel: *Ein Swapsatz, der interpoliert wurde, wird mit dem Wert '3' ('baInterestRateSwap' + 'baInterestRateInterpolated') gekennzeichnet.*

Geldmarkt- und Swapsätze können nicht gleichzeitig mit Zerosätzen verwendet werden.
Wie die verschiedenen Sätze in die Berechnung der Diskontfaktoren eingehen, ist im Kapitel „4.3.6 Calculate“ auf Seite 56 beschrieben.

4.2.12 Interpolation_Bootstrapping

Verwendung:	Interpolationsmethode für das Bootstrapping
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Für die Berechnung der Diskontfaktoren im Rahmen des Bootstrapping müssen ggf. Zinssätze interpoliert werden, wenn Laufzeiten benötigt werden, die nicht in der übergebenen Zinskurve enthalten sind. *Interpolation_Bootstrapping* legt fest, wie diese Werte interpoliert werden:

Konstante	Wert	Behandlung des Datums
baInterpolationCubicSpline	1	Interpolation mittels Cubic Spline
baInterpolationLinear	2	Lineare Interpolation

4.2.13 Interpolation_DiscountFactors

Verwendung:	Interpolationsmethode für Diskontfaktoren
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Nach erfolgreicher Berechnung der Diskontfaktoren liegen diese für alle in der Zinskurve enthaltenen (und ggf. interpolierten) Laufzeiten vor. *Interpolation_DiscountFactors* legt fest, wie Diskontfaktoren interpoliert werden sollen, die zwischen zwei bekannten Laufzeiten liegen:

Konstante	Wert	Behandlung des Datums
baInterpolationCubicSpline	1	Interpolation mittels Cubic Spline
baInterpolationLinear	2	Lineare Interpolation

4.2.14 Fixing

Verwendung:	Datum des Geschäftsabschlusses (Fixierung der Zinssätze)
Datentyp:	Date
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Fixing bezeichnet das Datum, an dem die Zinssätze der Zinskurve festgestellt bzw. vereinbart werden. Auf das Fixing folgt die Lieferung (Settlement). Informationen zum Lieferdatum und ein Beispiel finden Sie im Kapitel „4.2.17 Settlement“ auf Seite 50.

4.2.15 LastDate

Verwendung:	Letztes Datum der Zinskurve
Datentyp:	Date
Lese-/Schreibzugriff:	Ja/Nein

Nach erfolgreichem Aufruf der *Calculate*-Methode liefert *LastDate* das letzte Datum zurück, das von der Zinskurve abgedeckt wird (Settlement-Date + längste Laufzeit).

4.2.16 Calculated

Verwendung:	Rückmeldung, ob die Zinskurve berechnet wurde
Datentyp:	Boolean
Lese-/Schreibzugriff:	Ja/Nein

Calculated liefert den Wert '*True*' zurück, wenn der letzte Aufruf der Methode *Calculate* erfolgreich war und seitdem keine berechnungsrelevanten Eigenschaften verändert wurden.

4.2.17 Settlement

Verwendung:	Datum der Lieferung
Datentyp:	Date
Lese-/Schreibzugriff:	Ja/Nein

Settlement bezeichnet das Datum der Lieferung und Erfüllung des Geschäfts. Es wird berechnet aus *Fixing* und *SettlementDays*.

Beispiel: *Am 29.10. wird ein 6-Monats-Euribor von 4,598% p.a. auf der Reuters-Seite <EURIBOR01> veröffentlicht. Der 29.10. ist das Fixing-Date. Die Lieferung erfolgt usancegemäß 2 Bankarbeitstage (TARGET) nach dem Fixing-Date, also am 31.11. (Settlement-Date).*

4.2.18 SettlementDays

Verwendung:	Arbeitstage zwischen Fixing und Settlement
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

SettlementDays gibt die Arbeitstage an, die zwischen der Fixierung (*Fixing*) und der Lieferung (*Settlement*) liegen. Welche Tage als Arbeitstage gezählt werden, bestimmt sich nach den in *Calendar* definierten Feiertagen.

4.2.19 Spread

Verwendung:	Aufschlag auf alle Zinssätze
Datentyp:	Single
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Mit *Spread* kann eine Parallelverschiebung der Zinskurve durchgeführt werden. Für *Spread* kann ein Aufschlag (bzw. mit negativem Vorzeichen Abschlag) in der Form „P/100“ (4% = 4/100 = 0.04) angegeben werden, der bei der Berechnung der Diskontfaktoren berücksichtigt wird.

4.2.20 SwapFrequency

Verwendung:	Interpolations-/Zinszahlungsintervall der Swapsätze
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

SwapFrequency steuert, in welchem zeitlichen Intervall die Swapsätze interpoliert und im Rahmen des Bootstrapping angesetzt werden:

Konstante	Wert	Zeitintervall
baMonth	3	Monatlich
baQuarter	4	Quartalsweise
baYearSemi	5	Halbjährlich
baYear	6	Jährlich

4.2.21 SwapInterestMethod

Verwendung:	Verzinsungsmethode der Swapsätze
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

SwapInterestMethod legt fest, ob die Swapsätze im Bootstrapping einfach (simple) oder mit Zinseszins (compound) berechnet werden.

Konstante	Wert	Verzinsungsmethode

baSimple	1	Einfach
baCompound	2	Zinseszins

4.2.22 Tag

Verwendung:	Frei verwendbares Merkmal
Datentyp:	Variant
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

4.2.23 UserExitObject

Verwendung:	Verzweigung zu einem externen Zinskurven-Objekt
Datentyp:	Object
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Zinskurven spielen eine zentrale Rolle bei der Berechnung und Bewertung von Zahlungsströmen. Barwerte und Zinsterminsätze können sich bei unterschiedlichen Bootstrapping-Verfahren oder Methoden zur Interpolation von Zinssätzen und Diskontfaktoren deutlich unterscheiden. Wenn Sie die Funktionalitäten von babz verwenden möchten, dabei jedoch Ihre eigene Zinskurven-Methodik nutzen wollen, können Sie einen „User Exit“ programmieren. Dabei wird das Bootstrapping sowie die Rückmeldung von Diskontfaktoren und Zinsterminsätzen an ein externes Objekt ausgelagert:

1. Erstellen Sie eine Klasse, welche die Berechnung der Diskontfaktoren und Zinsterminsätze vornimmt. Diese Klasse muss über die folgenden Methoden und Eigenschaften verfügen:

   - **Eigenschaft UserExit_BAInterestCurve (Datentyp: Object)**

     Bevor Objekte der Klasse *BAInterestCurve* eine Funktion bzw. Methode des externen Zinskurven-Objekts aufrufen, erstellen sie für die Eigenschaft *UserExit_BAInterestCurve*

einen Verweis auf sich selbst. So kann das externe Zinskurven-Objekt auf alle Eigenschaften der Zinskurve zugreifen.

- **Methode UserExit_Bootstrapping(intInterestRates, intDiscountFactors)**
  Diese Methode wird gestartet, wenn für die Zinskurve die *Calculate*-Methode aufgerufen wird. Als Parameter werden die Aufzählungen *intInterestRates* (Zinssätze der Zinskurve, *Collection* mit Elementen des Typs *baInterestRate*) und *intDiscountFactors* (Diskontfaktoren der Zinskurve, *Collection* mit Elementen des Typs *baDiscountFactor*) übergeben. *UserExit_Bootstrapping* kann diese Aufzählungen verändern und mit Elementen der Typen *baInterestRate* bzw. *baDiscountFactor* befüllen. Nach Beendigung von *UserExit_Bootstrapping* werden diese Aufzählungen in den Eigenschaften *InterestRates* und *DiscountFactors* der Zinskurve hinterlegt.

  Vor dem Aufruf von *UserExit_Bootstrapping* wird das Settlement-Datum der Zinskurve berechnet, nach der Beendigung von *UserExit_Bootstrapping* wird die Eigenschaft *LastDate* berechnet.

- **Funktion UserExit_GetDiscountFactor(LookupDate)**
  Diese Funktion wird gestartet, wenn die Funktion *GetDF* der Zinskurve aufgerufen wird. Zurückzugeben ist der Diskontfaktor für das angegebene Datum als *Single*-Wert.

- **Funktion UserExit_GetZeroRate(LookupDate)**
  Diese Funktion wird gestartet, wenn die Funktion GetZeroRate der Zinskurve aufgerufen wird. Zurückzugeben ist die Zero Rate für das angegebene Datum als *Single*-Wert.

- **Funktion UserExit_GetForwardRate(Startdate, Enddate, DayCountConvention)**
  Diese Funktion wird gestartet, wenn die Funktion *GetForwardRate* der Zinskurve aufgerufen wird. Zurückzugeben ist der Zinsterminsatz (*Single*), wie er sich aus den Parametern *Startdate* und *Enddate* (beide Datentyp *Date*) sowie *DayCountConvention* (*Byte*) ergibt.

2. Fügen Sie jedem Objekt der Klasse *BAInterestCurve* über die Eigenschaft *UserExitObject* einen Verweis auf ein Objekt Ihrer Zinskurven-Klasse zu (dieser Verweis erfolgt nicht zentral für alle Objekte der Klasse *BAInterestCurve*, damit für die verschiedenen Zinskurven unterschiedliche Parameter und Methoden berücksichtigt werden können).

3. Stellen Sie die Systemeinstellung *InterestCurveUserExit* auf *True*. Alle Objekte der Klasse *BAInterestCurve* wenden dann den User Exit an.

4.3 Methoden

4.3.1 Clone

Funktion:	Erstellung einer identischen Kopie des Objekts
Parameter:	Keiner
Rückgabewert:	Objekt des Typs BAInterestCurve

Clone erstellt eine 100ige Kopie des Objekts und liefert einen Verweis auf diese Kopie zurück. Modifikationen an der Kopie wirken sich nicht auf das ursprüngliche Objekt aus.

4.3.2 GetDF

Funktion:	Rückgabe eines Diskontfaktors zu einem Datum
Parameter:	LookupDate (Date)
Rückgabewert:	Diskontfaktor zum Datum (Single)

Wurde die Zinskurve noch nicht vorbereitet, ruft *GetDF* automatisch die Methode *Calculate* auf. Muss der Diskontfaktor zum Datum *LookupDate* interpoliert werden, so erfolgt dies anhand der mit *Interpolation_DiscountFactors* festgelegten Methode. Der Zeitraum zwischen dem Settlement-Date und dem *LookupDate* wird für die Interpolation mit der Day Count Convention der Swapsätze ermittelt.

4.3.3 GetForwardCurve

Funktion:	Rückgabe einer impliziten Zinskurve
Parameter:	SettlementDate (Date)
Rückgabewert:	Implizite Zinsterminkurve (BAInterestCurve)

GetForwardCurve baut eine neue Zinskurve auf, deren Settlement auf *SettlementDate* gesetzt wird. *SettlementDate* muss nach dem Settlement der ursprünglichen Zinskurve liegen.

4.3.4 GetForwardRate

Funktion:	Rückgabe eines impliziten Zinsterminsatzes
Parameter:	StartDate (Date)
	EndDate (Date)
	DayCountConvention(Byte)
Rückgabewert:	Impliziter Zinsterminsatz (Single)

GetForwardRate berechnet aus der Zinskurve den impliziten Zinsterminsatz, der für die zukünftige Periode von *StartDate* bis *EndDate* gilt. *StartDate* darf nicht vor dem Settlement-Datum der Zinskurve liegen, *EndDate* nicht nach dem letzten Zinspunkt (*LastDate*).

Formeln & Berechnungsgrundlagen

Berechnung der Forward Rate

$$ForwardRate_{Startdate;Enddate} = \frac{\frac{Diskontfaktor_{Startdate}}{Diskontfaktor_{Enddate}} - 1}{DayCountFraction_{Startdate;Enddate;DayCountConvention}}$$

4.3.5 GetLastDate

Funktion:	Rückgabe des Datums des letzten Punktes auf der Zinskurve
Parameter:	Keiner
Rückgabewert:	Datum des letzten Punktes auf der Zinskurve (Date)

4.3.6 Calculate

Funktion:	Aufbereitung der Zinskurve für die weitere Verwendung
Parameter:	Keiner
Rückgabewert:	Keiner

Wenn alle Parameter der Zinskurve angegeben wurden, muss diese für die weitere Verwendung vorbereitet werden, d.h. aus der Zinskurve müssen die Diskontfaktoren berechnet werden. Dies geschieht mit der Methode *Calculate*.

Handelt es sich um eine Zinskurve aus Geldmarkt- und Swapsätzen, wird ein Bootstrapping durchlaufen:

Im Falle einer Zinskurve aus Geldmarkt- und Swapsätzen führt *Calculate* die folgenden Schritte durch:

1. Umrechung der Laufzeiten in die Einheit 'Jahre', soweit möglich.
2. Löschen unterjähriger Laufzeiten > 1 Jahr, da diese nicht quotiert werden.
3. Löschen interpolierter Zinssätze.
4. Berechnung von ***Date*** für alle Zinssätze, indem zum ***Settlement*** der Zinskurve die jeweilige Laufzeit addiert wird, abhängig von der Business Day Convention des Satzes.
5. Aufsteigende Sortierung der Zinssätze nach ***Date***.
6. Interpolation ggf. fehlender Swapsätze, so dass zwischen dem ersten und letzten angegebenen Swapsatz entsprechend des Intervalls ***SwapFrequency*** Swapsätze vorhanden sind. Dabei kommt die mit ***Interpolation_Bootstrapping*** definierte Interpolationsmethode zur Anwendung.

7. Berechnung der Diskontfaktoren für die Geldmarktsätze anhand der angegebenen Day Count Convention (***DayCountConvention_CashRates***). Für Geldmarktsätze werden keine zwischenzeitlichen Zinszahlungen angesetzt (d.h. Zinszahlung endfällig zum Datum ***Date***).
8. Berechnung der Diskontfaktoren für die Swapsätze anhand der angegebenen Day Count Convention (***DayCountConvention_SwapRates***). Es werden Zinszahlungen im Intervall ***SwapFrequency*** angesetzt.
9. Hinzufügen der interpolierten Zinssätze zur ***InterestRates***-Auflistung.
10. Hinzufügen der berechneten Diskontfaktoren zur ***DiscountFactors***-Auflistung.

Im Falle einer Zinskurve aus Zerosätzen werden die Schritte 6 bis 8 nicht durchgeführt. Die Diskontfaktoren werden direkt aus den angegebenen Sätzen berechnet, eine Interpolation von Zinssätzen ist nicht erforderlich.

Der *DiscountFactors*-Auflistung wird außer den aus den Zinssätzen berechneten Diskontfaktoren noch der Faktor '1,0' zum Settlement-Date der Zinskurve hinzugefügt.

Nach erfolgreichem Aufruf gibt die Eigenschaft *Calculated* den Wert '*True*' zurück.

Formeln & Berechnungsgrundlagen

Aufschlag des relativen und absoluten Spreads auf den Zinssatz

$$Rate = Zinssatz * \left(1 + Spread_{relativ)}\right) + Spread_{absolut}$$

Berechnung der Diskontfaktoren aus Cash Rates

$$df_t = \frac{1}{1 + Rate_t * DayCountFraction_t}$$

Berechnung der Diskontfaktoren aus Swap Rates(Simple)

$$df_t = \frac{1 - (\sum_{a=1}^{t-1} df_a * DayCountFraction_a) * Rate_t}{1 + Rate_t * DayCountFraction_t}$$

Berechnung der Diskontfaktoren aus Swap Rates(Compound)

$$df_t = \frac{1 - (\sum_{a=1}^{t-1} df_a^{\frac{1}{DayCountFraction_a}}) * Rate_t}{(1 + Rate_t)^{\frac{1}{DayCountFraction_t}}}$$

Berechnung der Diskontfaktoren aus Zero Rates

$$df_t = \frac{1}{(1 + Rate_t)^{DayCountFraction_t}}$$

Exkurs: Ermittlung von Diskontfaktoren aus Swapsätzen

Mit Hilfe von Diskontfaktoren soll der heutige Wert von zukünftigen Zahlungen ermittelt werden. Diskontfaktoren können nicht unmittelbar aus den in der Zinskurve angegebenen Swapsätzen abgeleitet werden, da diese Zinszahlungen in einem bestimmten Intervall (z.B. jährlich) implizieren, ein bestimmter Diskontfaktor jedoch eine Zahlung, die an einem zukünftigen Termin erfolgt, bewerten soll. Die Swapsätze müssen daher um die zwischenzeitlichen Zinszahlungen bereinigt werden, d.h. in Zero-Kupon-Sätze[2] umgerechnet werden. Dies geschieht im sog. Bootstrapping-Verfahren.

Im Folgenden sei i_n der Swapsatz und z_n der Zero-Kupon-Satz (Zero-Satz)für die Periode t_0 bis t_n. Der Zero-Satz z_1 entspreche dem Swapsatz i_1, da zwischen t_0 und t_1 keine Zinszahlungen geleistet

[2]Eine Nullkuponanleihe (Zero-Bond) zahlt bis zum Ende ihrer Laufzeit keine Zinsen. Die Zinsen werden am Ende der Laufzeit in einer Summe zusammen mit dem Nennbetrag gezahlt.

werden. Für t_2 kann der Zusammenhang zwischen Swapsatz und Zero-Satz wie folgt definiert werden:

$$1 = \frac{i_2}{(1+z_1)} + \frac{i_2}{(1+z_2)^2} + \frac{1}{(1+z_2)^2}$$

Diese Gleichung basiert auf der Prämisse, dass eine Anleihe, deren Kupon (i) dem Marktzins entspricht, zu pari notiert, der Preis der Anleihe also dem Nennwert entspricht. Der Preis ergibt sich aus der Summe der Barwerte aus den Zinszahlungen sowie dem Barwert der Rückzahlung. Die Gleichung kann nach z_2 aufgelöst werden. Ist z_2 bekannt, kann der Zero-Satz für t_3 berechnet werden, danach für t_4 usw. Allgemein für die Laufzeit t_n ergibt sich:

$$1 = \sum_{a=1}^{n-1} \frac{i_n}{(1+z_a)^a} + \frac{i_n + 1}{(1+z_n)^n}$$

aufgelöst nach z_n:

$$z_n = \sqrt[n]{\frac{i_n + 1}{1 - \sum_{a=1}^{n-1} \frac{i_a}{(1+z_a)^a}}}$$

Umgeschrieben in Diskontfaktoren anstelle von Zero-Sätzen und ergänzt um Day Count Fractions erhält man die oben unter „Berechnung von Diskontfaktoren aus Swap Rates“ aufgeführte Formel.

Hinweise zur Verwendung von Marktdaten

Achten Sie bei der Abbildung von Zinskurven und allgemein Marktdaten auf deren Eignung für die zu bearbeitende Aufgabe. So kann z.B. ein Swapsatz für die Bewertung eines Zinsswaps mit einem Zahlungsintervall von 6 Monaten geeignet sein, für einen Zinsswap mit einem Zahlungsintervall von 1 Monat hingegen nicht.

Grundsätzlich ist zu empfehlen, Zinskurven direkt mit Zero Rates abzubilden, sofern diese verfügbar sind. Dadurch entfällt die synthetische Ermittlung von Zero Rates (Bootstrapping), was die Genauigkeit und Vergleichbarkeit der Ergebnisse verbessern kann.

Weicht ein Berechnungsergebnis signifikant vom zu erwartenden Ergebnis ab und ist ein Fehler in der Abbildung des zu bewertenden Geschäftes und in der Übernahme der Marktdaten ausgeschlossen, prüfen Sie zunächst die Usancen der Marktdaten (insb. Day Count Convention). Lässt sich die Abweichung dadurch nicht erklären, kann der Vergleich der Marktdaten mit den von einem anderen Marktteilnehmer verwendeten Daten helfen (Datenquelle, Uhrzeit, …).

Bei Verdacht auf einen Fehler im babz-Rechenkern wenden Sie sich bitte an den Entwickler.

4.4 Ereignisse

4.4.1 AfterCalculation

Auslösung:	Nach erfolgreichem Abschluss der Methode Calculate
Parameter:	Keiner

4.4.2 PropertyChanged

Auslösung:	Nach Veränderung einer berechnungsrelevanten Eigenschaft
Parameter:	Keiner

5 Klasse BAFXForwardCurve

5.1 Aufgaben & Anwendung

Mit Objekten der Klasse *BAFXForwardCurve* werden Kassa- und Termin-Wechselkurse in Form einer Kurve abgebildet und für die Verwendung in anderen Klassen vorbereitet. Die zentrale Aufgabe ist die Bereitstellung von Kursen zur Bewertung von Devisengeschäften.

5.2 Eigenschaften

5.2.1 AskSpot

Verwendung:	Kassa-Briefkurs in Mengennotierung
Datentyp:	Single
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein
Mögliche Werte:	Positive Werte

AskSpot bezeichnet den Wechselkurs, zu dem die Fremdwährung (*CurrForeign*) gegen eine Einheit Heimwährung (*CurrDomestic*) in der Kasse verkauft werden kann. Wird *AskSpot* nicht angegeben, wird er bei Aufruf der Methode *Calculate* aus *MidSpot* und *BidSpot* berechnet, sofern diese angegeben sind.

5.2.2 BusinessDayConvention

Verwendung:	Business Day Convention der Terminsätze
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Die Business Day Convention wird benötigt, um ausgehend vom Settlement-Datum das Datum zu bestimmen, für das ein Termin-Wechselkurs gültig ist.

Details zu den Business Day Conventions finden Sie im Kapitel „3.3.11 ShiftDate“ auf Seite 40.

5.2.3 BidSpot

Verwendung:	Kassa-Geldkurs in Mengennotierung
Datentyp:	Single
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein
Mögliche Werte:	Positive Werte

BidSpot bezeichnet den Wechselkurs, zu dem die Fremdwährung (*CurrForeign*) gegen eine Einheit Heimwährung (*CurrDomestic*) in der Kasse gekauft werden kann. Wird *BidSpot* nicht angegeben, wird er bei Aufruf der Methode *Calculate* aus *MidSpot* und *AskSpot* berechnet, sofern diese angegeben sind.

5.2.4 Calendar

Verwendung:	Definition des Bankkalenders
Datentyp:	BACalendar
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Auf *Calendar* setzen alle Terminberechnungen (Ausrollen des Zeitgerüstes) der Klasse auf. Ein *BACalendar*-Objekt muss auch dann zugewiesen werden, wenn keine Feiertage definiert werden sollen.

5.2.5 CurrDomestic

Verwendung:	Heimwährung der Wechselkurse und Swapsätze der Kurve
Datentyp:	String
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

CurrDomestic bezeichnet die Heimwährung des Währungspaars, auf das sich alle Wechselkurse und Swapsätze der Kurve beziehen. Die Angabe der Fremdwährung erfolgt mit *CurrForeign*. Grundsätzlich kann die Bezeichnung der Währungen beliebig erfolgen (z.B. 'EUR', 'EURO' o.ä.), es wird jedoch empfohlen, einheitlich ISO 4217 anzuwenden (also 'EUR', 'USD' etc.).

5.2.6 CurrForeign

Verwendung:	Fremdwährung der Wechselkurse und Swapsätze der Kurve
Datentyp:	String
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

5.2.7 Description

Verwendung:	Erklärende Bezeichnung des Objekts
Datentyp:	String
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

5.2.8 Fixing

Verwendung:	Datum des Geschäftsabschlusses (Fixierung der Wechselkurse)
Datentyp:	Date
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Fixing bezeichnet das Datum, an dem die Wechselkurse und Swapsätze der Zinskurve festgestellt bzw. vereinbart werden. Auf das Fixing folgt die Lieferung (Settlement). Informationen zum Lieferdatum und ein Beispiel (für Zinssätze) finden Sie im Kapitel „4.2.17 Settlement“ auf Seite 50.

5.2.9 Interpolation

Verwendung:	Interpolationsmethode für Swapsätze
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Interpolation legt fest, wie Swapsätze interpoliert werden sollen, die zwischen zwei bekannten Laufzeiten liegen:

Konstante	Wert	Behandlung des Datums
baInterpolationCubicSpline	1	Interpolation mittels Cubic Spline
baInterpolationLinear	2	Lineare Interpolation

5.2.10 LastDate

Verwendung:	Letztes Datum der Terminkurve
Datentyp:	Date
Lese-/Schreibzugriff:	Ja/Nein

Nach erfolgreichem Aufruf der *Calculate*-Methode liefert *LastDate* das letzte Datum zurück, das von der Terminkurve abgedeckt wird (Settlement-Date + längste Laufzeit).

5.2.11 MidSpot

Verwendung:	Kassa-Mittelkurs in Mengennotierung
Datentyp:	Single
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein
Mögliche Werte:	Positive Werte

MidSpot ist der Mittelwert aus dem Kassa-Geldkurs (*BidSpot*) und dem Kassa-Briefkurs (*BidAsk*). Wird *MidSpot* nicht angegeben, wird er bei Aufruf der Methode *Calculate* aus *BidSpot* und *AskSpot* berechnet, sofern diese angegeben sind.

5.2.12 Outrights

Verwendung:	Aus Kassakursen und Swapsätzen berechnete Terminkurse
Datentyp:	Collection
Lese-/Schreibzugriff:	Ja/Nein
Mögliche Werte:	Werte des Typs baFXOutright

Nach erfolgreichem Aufruf von *Calculate* enthält die *Outrights*-Auflistung alle berechneten Devisen-Terminsätze. Die Outrights werden in Variablen des Typs *baFXOutright* gespeichert, der mit folgenden Elementen ausgestattet ist:

Element	Datentyp	Hinweis
Date	Date	Valutadatum
Bid	Single	Geldkurs
Ask	Single	Briefkurs
Mid	Single	Mittelkurs

Outrights können über die Funktion *GetOutright* abgefragt werden.

5.2.13 Calculated

Verwendung:	Rückmeldung, ob die Terminkurve berechnet wurde
Datentyp:	Boolean
Lese-/Schreibzugriff:	Ja/Nein

Calculated liefert den Wert '*True*' zurück, wenn der letzte Aufruf der Methode *Calculate* erfolgreich war und seitdem keine berechnungsrelevanten Eigenschaften verändert wurden.

5.2.14 Settlement

Verwendung:	Datum der Lieferung
Datentyp:	Date
Lese-/Schreibzugriff:	Ja/Nein

Settlement bezeichnet das Datum der Lieferung und Erfüllung des Kassa-Geschäfts. Es wird berechnet aus *Fixing* und *SettlementDays*.

5.2.15 SettlementDays

Verwendung:	Arbeitstage zwischen Fixing und Settlement
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

SettlementDays gibt die Arbeitstage an, die zwischen der Fixierung (*Fixing*) und der Lieferung des Kassageschäfts (*Settlement*) liegen. Welche Tage als Arbeitstage gezählt werden, bestimmt sich nach den in *Calendar* definierten Feiertagen.

5.2.16 Swaps

Verwendung:	Swapsätze zur Berechnung der Terminkurse
Datentyp:	Collection
Lese-/Schreibzugriff:	Ja/Ja
Mögliche Werte:	Werte des Typs baFXSwapRate

In der *Swaps*-Auflistung werden die Swapsätze definiert, anhand derer die Methode *Calculate* die Devisen-Terminkurse berechnet. Die Beschreibung der Swaps erfolgt anhand Variablen des Datentyps *baFXSwapRate*, der folgende Elemente enthält:

Element	Datentyp	Hinweis
Date	Date	Valutadatum
Tenor	Single	Laufzeit
TenorUnit	Byte	Einheit der Laufzeit
BidSwap	Single	Swapsatz Geldkurs
AskSwap	Single	Swapsatz Briefkurs

Tenor & TenorUnit

Mit *Tenor* und Tenor*Unit* wird definiert, auf welche Laufzeit sich der Swapsatz bezieht, d.h. wann das Lieferdatum für den jeweiligen Terminsatz erfolgt. Die Zeiteinheit wird mit Tenor*Unit* angegeben, die Anzahl der Zeiteinheiten mit *Tenor*. Folgende Zeiteinheiten stehen zur Verfügung:

Konstante	Wert	Zeitintervall
baDay	1	1 Tag
baWeek	2	1 Kalenderwoche
baMonth	3	1 Kalendermonat
baQuarter	4	3 Kalendermonate
baYearSemi	5	6 Kalendermonate
baYear	6	12 Kalendermonate

Date

In *Date* ist nach erfolgreichem Aufruf der *Calculate*-Methode das mittels *Tenor* und Tenor*Unit* berechnete Lieferdatum gespeichert (ausgehend von *Settlement*, unter Berücksichtigung der gültigen Business Day Convention).

BidSwap & AskSwap

BidSwap und *AskSwap* bezeichnen die Aufschläge (Report) und Abschläge (Deport) auf den Kassa-Geldkurs (*BidSpot*) bzw. Kassa-Briefkurs (*AskSpot*). Der Devisenterminkurs wird berechnet durch Addition des jeweiligen Kassakurses mit dem entsprechenden Swapsatz.

5.2.17 Tag

Verwendung:	Frei verwendbares Merkmal
Datentyp:	Variant
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

5.3 Methoden

5.3.1 GetOutright

Funktion:	Rückgabe eines Devisen-Terminsatzes zu einem Datum
Parameter:	LookupDate (Date)
	RateType (Byte)
Rückgabewert:	Devisen-Terminkurs (Single)

Mit der Funktion *GetOutright* kann für ein Valutadatum *LookupDate* ein Devisen-Terminkurs (Outright) abgefragt werden. *LookupDate* muss zwischen dem Settlement-Datum (*Settlement*) und dem letzten durch die Kurve abgedeckten Swapsatz liegen (*LastDate*), eine Extrapolation erfolgt nicht.

Mit *RateType* wird festgelegt, ob der Briefkurs, Geldkurs oder Mittelkurs zurückgegeben wird:

Konstante	Wert	Kurstyp
baBid	1	Geldkurs
baAsk	2	Briefkurs
baMid	3	Mittelkurs

Wurde die Kurve noch nicht vorbereitet (*Calculated* hat den Wert 'False'), wird automatisch die Methode *Calculate* aufgerufen. Interpolationen zwischen zwei Stützstellen (die durch die Swapsätze bestimmt sind) erfolgen anhand der in *Interpolation* eingestellten Methode.

5.3.2 Calculate

Funktion:	Aufbereitung der Kurve für die weitere Verwendung
Parameter:	Keiner
Rückgabewert:	Keiner

Calculate berechnet aus den Swapsätzen die Devisenterminkurse und speichert diese in der Auflistung *Outrights*. Nach erfolgreichem Aufruf hat die *Calculated*-Eigenschaft den Wert 'True' und die Kurve kann für weitere Berechnungen verwendet werden.

5.4 Ereignisse

5.4.1 AfterCalculation

Auslösung:	Nach erfolgreichem Abschluss der Methode Calculate
Parameter:	Keiner

5.4.2 PropertyChanged

Auslösung:	Nach Veränderung einer berechnungsrelevanten Eigenschaft
Parameter:	Keiner

6 Klasse BAOutput

6.1 Aufgaben und Anwendung

Objekte der Klasse *BAOutput* dienen der Aufnahme, Weiterverarbeitung und Auswertung von Daten. Beispielsweise schreiben Objekte der Klasse *BALoan* ihre Berechnungsergebnisse in ein *BAOutput*-Objekt, von wo aus diese z.B. für Barwertberechnungen herangezogen werden können.

6.2 Eigenschaften

6.2.1 Description

Verwendung:	Erklärende Bezeichnung des Objekts
Datentyp:	String
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein
Mögliche Werte:	Keine Beschränkung

6.2.2 DataArray

Verwendung:	Speicherung von Daten
Datentyp:	baDataArray
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja
Mögliche Werte:	Entsprechend der Spezifikationen von baDataArray

Variablen des Datentyps *baDataArray* beinhalten lediglich die beiden Elemente *Header* und *Data*. Bei *Data* handelt es sich um einen zweidimensionalen Array, in dem die einzelnen Daten matrixförmig gespeichert werden. Auf jedes Datum kann in der Form *Data(Y, X)* zugegriffen werden. *Header* ist ein eindimensionaler Array, in dem gespeichert wird, in welcher Spalte von *Data* sich welcher Datentyp befindet.

Header	$Datentyp_1$	$Datentyp_2$	$Datentyp_3$	$Datentyp_n$

Data	X_1	X_2	X_3	X_n
Y_1	$Datum_{1,1}$	$Datum_{1,2}$	$Datum_{1,3}$	$Datum_{1,n}$
Y_2	$Datum_{2,1}$	$Datum_{2,2}$	$Datum_{2,3}$	$Datum_{2,n}$
Y_3	$Datum_{3,1}$	$Datum_{3,2}$	$Datum_{3,3}$	$Datum_{3,n}$
Y_m	$Datum_{m,1}$	$Datum_{m,2}$	$Datum_{m,3}$	$Datum_{m,n}$

Welche Datentypen in einer Variable des Typs *baDataArray* gespeichert werden, hängt von der Klasse ab, welche die Variable befüllt.

6.2.3 DateSum

Verwendung:	Zeitintervalle zur Berechnung von Summen
Datentyp:	Collection
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein
Mögliche Werte:	Werte vom Typ baDateSum

Mit der *DateSum-Eigenschaft* können Sie für für in *DataArray* enthaltene Daten Summen über frei definierbare Zeitintervalle berechnen. Dazu wird die DateSum-Auflistung mit Variablen des Typs *baDateSum* gefüllt, die jeweils angeben, für welches Intervall und welchen Datentyp eine Summe zu bilden ist. Nach Aufruf der Methode *GetDateSum* enthalten die einzelnen Elemente die Berechnungsergebnisse.

Elemente des Datentyps *baDateSum* und deren Verwendung:

Element	Datentyp	Hinweis
Description	String	Bezeichnung
DataColumn	Byte	In DataArray enthaltener Datentyp, über den die Summe gebildet werden soll.
Start	Variant	Datum, zu dem die Summenbildung beginnen soll (inklusive)
Stop	Variant	Datum, zu dem die Summenbildung enden soll (inklusive)
Sum	Double	Summe (Berechnungsergebnis)
Type	Byte	Typ des Eintrags (s.u.)

Für die Elemente *Start* und *Stop* kann ein absolutes Datum angegeben werden (Datentyp *Date*) oder eine Datumsdefinition in Form einer Zeichenkette:

Zeichen	Mögliche Werte	Hinweis
1	F, S oder T	Bestimmt, ob das Datum anhand des Fixing-Datums der Zinskurve (F), des Settlement-Datums der Zinskurve (S) oder des tagesaktuellen Datums (T) berechnet wird
2	+	
3 – N	Intervall-Zähler	Bestimmt, wie oft das im nächsten Block definierte Zeitintervall zum Basisdatum addiert wird.
N+1	D, W, M oder Y	Legt fest, welches Zeitintervall zur Anwendung kommt: Tage (D), Wochen (W), Monate (M) oder Jahre (Y).
N+2	E	Optionaler Wert: Gibt an, ob der berechnete Termin exklusiv oder inklusiv bewertet werden soll.

Beispiel: *Ist das Fixing-Datum der Zinskurve der 01.01.2000 und für 'Start' wird die Zeichenkette 'F+1Y' angegeben, werden für eine Summenbildung solche Elemente*

berücksichtigt, die am oder nach dem 01.01.2001 auftreten. Die Angabe 'F+1YE' berücksichtigt nur Elemente nach dem 01.01.2001.

Das Element *Type* kann folgende Werte annehmen:

Konstante	Wert	Hinweis
baNone	0	Benutzerdefiniertes Zeitintervall
baDateSumTotal	1	Summe über alle Elemente des Datentyps
baDateSumRequestedPeriods	2	Summe über alle benutzerdefinierten Zeitintervalle des Datentyps
baDateSumSpecifiedPeriod	3	Summe über die gesamte durch die benutzerdefinierten Zeitintervalle aufgespannte Zeitspanne

Fügen Sie der *DateSum-Collection* nur Elemente an, deren Feld *Type* auf *baNone* gesetzt ist, nur diese werden bei der Summenbildung berücksichtigt. Elemente der Typen *baDateSumTotal*, *baDateSumRequestedPeriods* und *baDateSumSpecifiedPeriod* werden der Auflistung automatisch hinzugefügt. Anhand der verschiedenen Typen kann z.B. überprüft werden, ob die definierten Zeitintervalle alle Daten umfassen.

Beispiel: *In einem DataArray ist ein Datentyp enthalten, dessen erstes Element am 01.01.2000 und dessen letztes Element am 31.12.2010 auftritt. Für diesen Datentyp wurden zwei benutzerdefinierte Zeitintervalle (Type 0) angegeben, 01.01.2002 bis 31.12.2002 und 01.01.2004 bis 31.12.2004. Nach dem Aufruf von GetDateSum enthält die DateSum-Auflistung folgende Elemente:*

1. Type baNone

Die Summen über die beiden benutzerdefinierten Intervalle (01.01.2002 bis 31.12.2002 und 01.01.2004 bis 31.12.2004).

2. Type baDateSumTotal

Die Summe über alle Elemente des Datentyps (01.01.2000 bis 31.12.2010).

3. Type baDateSumRequestedPeriods

Die Summe über die beiden benutzerdefinierten Zeitintervalle (01.01.2002 bis 31.12.2002 + 01.01.2004 bis 31.12.2004).

4. Type baDateSumSpecifiedPeriod

Die Summe über das Zeitintervall, dessen Beginn vom frühesten Beginn und dessen Ende vom spätesten Ende eines benutzerdefinierten Zeitintervalls bestimmt wird (01.01.2002 bis 31.12.2004).

6.2.4 InterestCurve

Verwendung:	Zinskurve zur Berechnung von Barwerten
Datentyp:	BAInterestCurve
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein
Mögliche Werte:	Gültiges Objekt der Klasse BAInterestCurve

Anhand der in *InterestCurve* gespeicherten Zinskurve berechnet die Methode *GetPresentValue* Barwerte von in *DataArray* enthaltenen Daten.

6.2.5 PresentValue

Verwendung:	Rückgabe und Speicherung einer Barwert-Berechnung
Datentyp:	baPresentValue
Lese-/Schreibzugriff:	Ja/Nein
Eingabe erforderlich:	Nein
Mögliche Werte:	Werte des Typs baPresentValue

Nach dem Aufruf der *GetPresentValue*-Funktion wird das Berechnungsergebnis (zusätzlich zum Rückgabewert) in *PresentValue* gespeichert.

6.2.6 Tag

Verwendung:	Frei verwendbares Merkmal
Datentyp:	Variant
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein
Mögliche Werte:	Keine Beschränkung

6.3 Methoden

6.3.1 ConvertPresentValue

Funktion:	Rückgabe des Inhalts von PresentValue als Objekt der Klasse BAOutput
Parameter:	Keine
Rückgabewert:	Objekt der Klasse BAOutput

Ein in *PresentValue* gespeicherter Barwert wird als Objekt der Klasse *BAOutput* zurückgegeben, womit auf ihn alle Operationen angewandt werden können, die für die Klasse *BAOutput* zur Verfügung stehen. Die Eigenschaften *DateSum*, *InterestCurve* und *Tag* werden von dem *BAOutput*-Objekt, für das *ConvertPresentValue* aufgerufen wird, auf das zurückgegebene Objekt übertragen.

6.3.2 GetColumn

Funktion:	Ermittelt eine Spalte in DataArray
Parameter:	DataHeader (Byte)
Rückgabewert:	Ermittelte Spalte (Byte)

GetColumn gibt die Spalte in *DataArray.Data* zurück, in welcher der Datentyp *DataHeader* gespeichert ist.

6.3.3 GetDateSum

Funktion:	Berechnet Summen über Zeitintervalle
Parameter:	Keine
Rückgabewert:	Keiner

GetDateSum verarbeitet die in der *DateSum-Auflistung* definierten Zeitintervalle und gibt die Berechnungsergebnisse in die Auflistung aus. Details finden Sie im Kapitel DateSum (S. 72).

6.3.4 GetPresentValue

Funktion:	Ermittlung eines Barwertes
Parameter:	DataColumn (Byte)
	Invert (Boolean), optional
Rückgabewert:	Berechnungsergebnis (baPresentValue)

Für den in *DataColumn* angegebenen Datentyp wird der Barwert auf das Settlement-Date der Zinskurve berechnet, die in *InterestCurve* definiert ist. Das Ergebnis wird als Variable des Datentyps *baPresentValue* zurückgegeben und parallel in *PresentValue* gespeichert.

Wird als Parameter *Invert* auf '*False*' gesetzt, werden die Vorzeichen der Beträge umgekehrt (Multiplikation mit '-1').

6.3.5 GetRow

Funktion:	Ermittlung einer Zeile in DataArray
Parameter:	LookupDate (Date)
	DataColumn (Byte, optional)
Rückgabewert:	Ermittelte Zeile (Integer)

GetRow gibt die Zeile in *DataArray.Data* zurück, in welcher Daten zum Termin *LookupDate* gespeichert sind. Hierfür wird der Datentyp *baHeadDate* ausgewertet, wenn nicht mit

DataColumn ein anderer Datentyp angegeben wird (z.B. *baHeadValueDate*, *baHeadFixingDate*).

6.3.6 Merge

Funktion:	Zusammenführung von zwei Objekten des Typs BAOutput
Parameter:	MergeOutputObject (BAOutput)
Rückgabewert:	Zusammengeführtes Objekt (BAOutput)

Mit *Merge* kann ein Objekt des Datentyps *BAOutput* mit einem weiteren Objekt desselben Datentyps zusammengeführt werden. Das kann dann sinnvoll sein, wenn in separaten *BAOutput*-Objekten gespeicherte Berechnungsergebnisse gemeinsam weiterverarbeitet werden sollen. Für die Zusammenführung gilt:

Als Kriterium für die Sortierung und Bildung von Zwischensummen wird *baHeadDate* herangezogen, sofern dieser Datentyp in beiden eingehenden Objekten enthalten ist. Andernfalls wird *baHeadValueDate* verwendet, wenn dieser Datentyp in beiden Objekten enthalten ist. Kommen weder *baHeadDate* noch *baHeadValueDate* in Betracht, wird die Zusammenführung abgebrochen.

Es werden nur solche Datentypen in die Zusammenführung einbezogen, die einen Betrag darstellen und somit addiert werden können:

- baHeadDrawing
- baHeadInterestCalculation
- baHeadInterestPayment
- baHeadAccruedInterest
- baHeadAmortisationCalculation
- baHeadAmortisationPayment
- baHeadAccruedAmortisation
- baHeadBalance

- baHeadCashflow
- baHeadMarginCalculation
- baHeadMarginPayment
- baHeadAccruedMargin
- baHeadPresentValue
- baHeadCostOfAdministration
- baHeadCostOfRisk
- baHeadProfitLoss
- baHeadDisagio
- baHeadAmountDomestic
- baHeadAmountForeign

Merge prüft nicht, ob die Währungen der beiden eingehenden Objekte übereinstimmen. Stellen Sie vor dem Aufruf sicher, dass nur Beträge mit identischen Währungen addiert werden.

Beim zusammengeführten Objekt sind nur die Eigenschaften *Description* und *DataArray* gefüllt. Für die übrigen Eigenschaften kann *Merge* nicht entscheiden, ob sie auf das zusammenführte Objekt Anwendung finden.

7 Klasse BALoan

7.1 Aufgaben & Anwendung

Die Klasse *BALoan* berechnet anhand verschiedener Parameter Bestands- und Bewegungsgrößen von Darlehen. Die berechneten Daten werden als Klasse *BAOutput* zurückgegeben und können so für weitere Anwendungen herangezogen werden (z.B. Einstandsermittlung, Vorfälligkeitsentschädigung, Barwerte).

7.2 Übergeordnete Datentypen

7.2.1 baDateControl

Der Datentyp *baDateControl* dient der Definition von Datenreihen. Er kommt bei den Eigenschaften *Amortisation* und *Interest* der Klasse *BALoan* zum Einsatz und verfügt über folgende Elemente:

Element	Datentyp	Hinweis
First	Date	Erste Ausführung
Interval	Integer	Ausführungsintervall in Monaten
BusinessDayConvention	Byte	Bei der Terminberechnung anzuwendende Business Day Convention
Count	Integer	Anzahl der Ausführungen
Last	Date	Letzte Ausführung (wird berechnet)
Ultimo	Boolean	Gibt an, dass die Termine stets auf den letzten Kalendertag des jeweiligen Monats fallen

Die erste Ausführung erfolgt zum mit *First* angegebenen Datum. Danach erfolgt die Anzahl im angegebenen Intervall. Fällt *First* auf den letzten Kalendertag eines Monats und sollen die nachfolgenden Termine ebenfalls auf den letzten Kalendertag des jeweiligen Monats fallen, muss *Ultimo* auf '*True*' gesetzt werden. Andernfalls werden die Termine auf denselben Kalendertag gesetzt wie in *First* definiert (ohne jedoch den jeweiligen Monat zu verlassen).

Beispiel: *Die erste Ausführung erfolgt am letzten Tag des Monats Februar ('First' = '28.02.2009'), danach monatlich. Damit die Ausführung an den Folgeterminen ebenfalls zum Monatsletzten erfolgt, wird 'Ultimo' auf 'True' gesetzt. Der folgende Termin ist dann der '31.03.2009'. Wird 'Ultimo' auf 'False' gesetzt, wird als Folgetermin der '28.03.2009' berechnet.*

Die Ausführung erfolgt insgesamt (d.h. einschließlich der ersten Ausführung) so oft, wie in *Count* angegeben. *Last* wird automatisch aus den vorgenannten Parametern berechnet. Wird für *Count* der Wert *'0'* übergeben, wird *Count* aus den Werten *First* und *Last* berechnet. *Last* wird so angepasst, dass es konsistent zu *First* ist (Kalendertag, Ultimo). *First* kann hingegen nicht aus *Count* und Last berechnet werden.

Fällt ein Datum auf einen Bankfeiertag des für das jeweilige Objekt der Klasse *BALoan* angegebenen Kalenders, wird es nach den in *BusinessDayConvention* angegebenen Regeln verschoben. Details finden Sie im Kapitel „3.3.11 ShiftDate“ auf Seite 40.

7.3 Eigenschaften

7.3.1 Amortisation

Verwendung:	Definition der Tilgung des Darlehens
Datentyp:	baAmortisation
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja
Mögliche Werte:	Entsprechend der Spezifikationen von baAmortisation

Mit der Eigenschaft Amortisation wird festgelegt, wann und auf welche Weise ein Darlehen getilgt wird. Der Datentyp *baAmortisation* verfügt dazu über folgende Elemente:

Element	Datentyp	Hinweis
Dates	baDateControl	Erste Ausführung, Intervall und Anzahl der Ausführungen
Instalment	Double	Tilgungsrate (bei Annuitätendarlehen: incl. Zinsanteil)
NoAutomaticRate	Boolean	Gibt an, dass keine automatische Berechnung der Tilungsrate erfolgen soll
Type	Date	Art der Tilgung (Raten oder Annuitäten)
Prepayment	Date	Vorfällige Rückzahlung
InAdvance	Byte	Vorschüssige Zahlweise

Dates

Dates steuert mit Hilfe des Datentyps *baDateControl*, wann die Tilgungsraten angerechnet und gezahlt werden. Die Anrechnung bezeichnet, wann zinswirksam die Reduzierung des Kreditsaldos erfolgt. Die Zahlung gibt an, wann die Auszahlung der Tilgungsleistung erfolgt. Die in *Dates* bezeichneten Termine definieren stets die Zahlungstermine.

Die Anrechnungstermine bestimmen sich danach, ob die Anrechnung „Adjusted“ oder „Unadjusted“ erfolgt. Dies wird dadurch gesteuert, ob dem Element *BusinessDayConvention* die Konstante *baAdjusted* hinzugefügt wurde oder nicht.

- **Unadjusted (Standard)**

  Fällt das berechnete (Zahlungs-)Datum auf einen Feiertag, wird es nach den Regeln der angegebenen Business Day Convention auf einen Arbeitstag verschoben (z.B. zwei Kalendertage nach hinten). Das Anrechnungsdatum wird jedoch nicht verschoben, d.h. Anrechnungs- und Zahlungstermin fallen auseinander (2 Tage). Die Zinsen der Zeitspanne zwischen Anrechnung und Zahlung werden erst in der nachfolgenden Periode berücksichtigt.

- **Adjusted**

  Das Anrechnungsdatum wird mit dem Zahlungsdatum verschoben, die Zinsen für den Zeitraum, um den das Zahlungsdatum gegenüber dem ursprünglichen Datum verschoben wurde, werden direkt mit der laufenden Zahlung berücksichtigt (nicht wie bei Unadjusted in der Folgeperiode).

Instalment

Instalment gibt die Tilgungsrate in Geldeinheiten an. Die Tilgungsrate wird mit negativem Vorzeichen geführt (aus der Sicht des Darlehensnehmers). Bei Annuitätendarlehen beinhaltet die Tilgungsrate Zins- und Tilgungsleistungen. Die Tilgungsrate wird automatisch berechnet, sofern nicht *NoAutomaticRate* auf '*True*' gesetzt ist. Eine ggf. vor Ende der planmäßigen Laufzeit des Darlehens gesetzte vorfällige Rückzahlung des Darlehens hat keinen Einfluss auf die automatisch berechneten Tilgungsraten.

NoAutomaticRate

Ist *NoAutomaticRate* auf '*False*' gesetzt (Standardwert), wird die Tilgungsrate (*Instalment*) automatisch so festgelegt, dass das Darlehen planmäßig auf den vorgegebenen Restsaldo zurückgeführt wird. Ist *NoAutomaticRate* hingegen auf '*True*' gesetzt, wird die angegebene Rate herangezogen.

Type

Type legt fest, ob die Tilgung des Darlehens ratierlich oder annuitätisch erfolgt.

Konstante	Wert	Hinweis
baInstalment	1	Konstante Tilgungsleistung, abnehmender Cashflow
baAnnuity	2	Konstanter Cashflow, zunehmende Tilgungsleistung

Prepayment

Für *Prepayment* kann ein Datum vor planmäßigem Ende des Darlehens angegeben werden, zu dem eine vollständige Rückführung des zu diesem Zeitpunkt noch offenen Kreditsaldos und der bis zu diesem Tag ggf. aufgelaufenen Zinsen erfolgt. Es werden nur solche Termine berücksichtigt, die nach dem Ziehungstermin liegen.

InAdvance

InAdvance gibt an, ob Zahlungen vorschüssig geleistet werden, was nur bei Annuitätendarlehen eine Rolle spielt. Ist *InAdvance* auf '*False*' gesetzt (Standardwert), so erfolgt die Zahlung der Zins-

und Tilgungsleistung jeweils am Ende der Zins- und Tilgungsperioden. Ist *InAdvance* auf '*True*' gesetzt, erfolgt die Zahlung der Zins- und Tilgungsleistung jeweils am Beginn der Zins- und Tilgungsperioden (vorschüssig). Es werden jedoch stets nur die bis zum Zahlungszeitpunkt aufgelaufenen Zinsen gezahlt, es werden keine Zinsen für nachfolgende Perioden in Ansatz gebracht (kein „In Area“).

7.3.2 Amount

Verwendung:	Angabe des Kreditbetrags
Datentyp:	Double
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein
Mögliche Werte:	Positive Werte

Amount bezeichnet den Kreditbetrag einschließlich eines ggf. anzusetzenden Disagios. Der Kreditbetrag unterscheidet sich damit vom Ziehungsbetrag, der kein Disagio enthält. Sie können entweder den Kreditbetrag angeben oder den Ziehungsbetrag, der jeweils andere Werte wird automatisch unter Berücksichtigung eines Disagios berechnet. Werden beide Werte angegeben, überschreibt der Kreditbetrag den Ziehungsbetrag.

Details zum Ziehungsbetrag finden Sie im Kapitel „7.3.8 Drawing“ auf Seite 86.

7.3.3 Calculated

Verwendung:	Rückmeldung, ob das Darlehen berechnet wurde
Datentyp:	Boolean
Lese-/Schreibzugriff:	Ja/Nein

Calculated liefert den Wert '*True*' zurück, wenn der letzte Aufruf der Methode *Calculate* erfolgreich war und seitdem keine berechnungsrelevanten Eigenschaften verändert wurden.

Spiegelt das Berechnungsergebnis (*Output*) nicht die Eigenschaften wider, liefert *Calculated* '*False*' zurück.

7.3.4 Calendar

Verwendung:	Definition des Bankkalenders
Datentyp:	BACalendar
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Auf *Calendar* setzen alle Terminberechnungen (Ausrollen des Zeitgerüstes, Zinstage etc.) der Klasse auf. Ein *BACalendar*-Objekt muss auch dann zugewiesen werden, wenn keine Feiertage definiert werden sollen.

7.3.5 Curr

Verwendung:	Währung des Darlehens
Datentyp:	String
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Abhängig von Systemeinstellung CurrRequired

Curr legt die Währung des Darlehens fest. Es können nur solche Objekte (z.B. Zinskurven) auf das Darlehen angewandt werden, welche dieselbe Währung aufweisen. Ist keine Währung angegeben, findet keine Kontrolle statt.

Ist die Systemeinstellung *CurrRequired* explizit auf *True* gesetzt, muss eine Währung angegeben werden. Nähere Informationen finden Sie im Abschnitt „2.2.3 CurrRequired“ auf Seite 20.

7.3.6 Description

Verwendung:	Erklärende Bezeichnung des Objekts
Datentyp:	String
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein
Mögliche Werte:	Keine Beschränkung

7.3.7 Disagio

Verwendung:	Angabe eines Disagios
Datentyp:	Single
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein
Mögliche Werte:	Zwischen '0' und '1'

Das Disagio wird als Prozentsatz in der Form „P/100“ angegeben (4% = 4/100 = 0.04).

7.3.8 Drawing

Verwendung:	Festlegung der Ziehung des Darlehensbetrages
Datentyp:	baDrawing
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja
Mögliche Werte:	Entsprechend der Spezifikationen von baDrawing

In der aktuellen Programmversion wird nur ein Ziehungstermin unterstützt. Dieser wird als Variable des Datentyps *baDrawing* definiert, der folgende Elemente beinhaltet:

Element	Datentyp	Hinweis
Date	Date	Datum, zu dem die Ziehung erfolgen soll.
Amount	Double	Erste Zinsanpassung

Element	Datentyp	Hinweis
BusinessDayConvention	Byte	Bei der Terminberechnung anzuwendende Business Day Convention.

Die Ziehung erfolgt zum Datum *Date*, das, sofern es nicht auf einen Bankarbeitstag fällt, nach der angegebenen Regel (*BusinessDayConvention*) verschoben wird. *Amount* bezeichnet den Ziehungsbetrag, der dem Kreditnehmer zufließt (Kreditbetrag ./. Disagio). Sie können entweder den Kreditbetrag angeben oder den Ziehungsbetrag, der jeweils andere Werte wird automatisch unter Berücksichtigung eines Disagios berechnet. Werden beide Werte angegeben, überschreibt der Kreditbetrag den Ziehungsbetrag.

Details zum Kreditbetrag finden Sie im Kapitel „7.3.2 Amount“ auf Seite 84.

7.3.9 Interest

Verwendung:	Definition der Zinskonditionen des Darlehens
Datentyp:	baInterest
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Mit der Eigenschaft *Interest* wird festgelegt, wann Zinszahlungen zu leisten sind und wie diese berechnet werden. Der Datentyp *baInterest* verfügt dazu über folgende Elemente:

Element	Datentyp	Hinweis
Dates	baDateControl	Erste Ausführung, Intervall und Anzahl der Ausführungen
NominalRate	Single	Nominalzinssatz p.a.
Margin	Single	Zinsmarge p.a., die im Nominalzinssatz enthalten ist
InterestMethod	Byte	Art der Verzinsung (einfach oder Zinseszins)
DayCountConvention	Byte	Day Count Convention
PaymentMode	Byte	Zahlweise bei Annuitätendarlehen
Float	baInterestFloat	Variable Verzinsung

Dates

Dates steuert mit Hilfe des Datentyps *baDateControl*, wann die Zinsleistungen angerechnet und gezahlt werden. Die Anrechnung bezeichnet, wann die Berechnung der Zinsleistung erfolgt. Die Zahlung gibt an, wann die Auszahlung der Zinsleistung erfolgt. Die in *Dates* bezeichneten Termine definieren stets die Zahlungstermine.

Die Anrechnungstermine bestimmen sich danach, ob die Anrechnung „Adjusted“ oder „Unadjusted“ erfolgt. Dies wird dadurch gesteuert, ob dem Element *BusinessDayConvention* die Konstante *baAdjusted* hinzugefügt wurde oder nicht.

- **Unadjusted (Standard)**

  Fällt das berechnete (Zahlungs-)Datum auf einen Feiertag, wird es nach den Regeln der angegebenen Business Day Convention auf einen Arbeitstag verschoben (z.B. zwei Kalendertage nach hinten). Das Anrechnungsdatum wird jedoch nicht verschoben, d.h. Anrechnungs- und Zahlungstermin fallen auseinander (2 Tage). Die Zinsen der Zeitspanne zwischen Anrechnung und Zahlung werden erst in der nachfolgenden Periode berücksichtigt.

- **Adjusted**

  Das Anrechnungsdatum wird mit dem Zahlungsdatum verschoben, die Zinsen für den Zeitraum, um den das Zahlungsdatum gegenüber dem ursprünglichen Datum verschoben wurde, werden direkt mit der laufenden Zahlung berücksichtigt (nicht wie bei Unadjusted in der Folgeperiode).

Bei jeder Tilgungsanrechnung wird automatisch auch eine Zinsanrechnung durchgeführt.

NominalRate

NominalRate gibt den Nominalzinssatz p.a. an, mit dem das Darlehen zu verzinsen ist. Der Wert muss in der Form „P/100“ vorliegen (4% = 4/100 = 0.04).

Margin

Margin gibt an, welche Zinsmarge (Bankenmarge) ggf. in *NominalRate* enthalten ist und wird ebenfalls in der Form „P/100“ angegeben.

DayCountConvention

Die Day Count Convention gibt an, wie Zinstage berechnet werden. Unterstützt werden die Usancen 30/360, 30/360, Act/360, Act/365 und Act/Act. Details finden Sie im Kapitel „3.3.5 GetDayCountFraction„ auf Seite 36.

PaymentMode

Bei Annuitätendarlehen kann es vorkommen, dass die Zeitspanne zwischen Kreditziehung und erster Zins- und Tilgungsleistung länger ist als die Zeitspannen zwischen den einzelnen Zins- und Tilgungsterminen. Der auf die übersteigende Zeitspanne entfallende Zinsanteil reduziert die anfänglichen Tilgungsanteile, so dass das Annuitätendarlehen „nicht aufgeht“ und am Ende ein vom gewünschten Restsaldo abweichender Saldo übrig bleibt. Mit diesem Zinsanteil kann wie folgt verfahren werden:

Konstante	**Wert**	**Hinweis**
baInterestUpfront	1	Die Zinsen werden zu Beginn der ersten Zinsperiode bezahlt.
baInterestFirstAnnuity	2	Die Zinsen werden zusammen mit der ersten Annuität bezahlt. Eine Aufzinsung erfolgt nicht.
baInterestCapitalise	3	Die Zinsen werden zu Beginn der ersten Zinsperiode kapitalisiert und die Annuität neu berechnet.

Beispiel: *Die Ziehung eines Darlehens erfolgt am 01.01.2008. Die Zahlung der ersten Annuität erfolgt am 31.01.2009, danach monatlich. Für den Zeitraum 01.01.2008 – 31.12.2008 sind € 1.000 Zinsen aufgelaufen, die in der Annuität nicht*

berücksichtigt sind, da in dieser nur Zinsen eingerechnet sind, die ab dem 01.01.2009 entstehen.

'baInterestUpfront'
Die Zinsen i.H.v. € 1.000 werden am 31.12.2008 zur Zahlung angesetzt.

'baInterestFirstAnnuity'
Die Zinsen i.H.v. € 1.000 werden am 31.01.2009 zusätzlich zu ersten Annuität zur Zahlung angesetzt.

'baInterestCapitalise'
Die Zinsen i.H.v. € 1.000 werden am 31.12.2008 kapitalisiert, d.h. der Kreditsaldo wird um die Zinsen erhöht. Anschließend wird die Annuität an den neuen Kreditsaldo angepasst.

Float

Mit *Float* kann angegeben werden, ob das Darlehen variabel verzinst werden soll. Die erforderlichen Parameter werden mit einer Variablen des Datentyps *baInterestFloat* festgelegt.

Element	Datentyp	Hinweis
InterestCurve	BAInterestCurve	Zinskurve, anhand derer die Zinssätze berechnet werden sollen.
First	Integer	Erste Zinsanpassung
AutoFloat	Boolean	Automatische Steuerung der Zinsanpassung
Interval	Byte	Anpassungsintervall
Enabled	Boolean	Variable Verzinsung aktivieren oder deaktivieren
CalculateAmortisation	Boolean	Automatische Neuberechnung der Rate bei Annuitätendarlehen
CalculateInitalInterestRate	Boolean	Berechnung des Satzes der ersten Zinsperiode

InterestCurve

InterestCurve gibt die Zinskurve an, anhand derer die Zinssätze des Darlehens berechnet werden sollen. Die Zinskurve muss den vom Darlehen in Anspruch genommenen Zeitraum abdecken, d.h. ihr Settlement-Datum darf nicht nach dem Beginn der ersten Zinsperiode liegen, für die ein Zinssatz berechnet werden soll. Das letzte Datum (*LastDate*) darf nicht vor dem Ende der letzten Zinsperiode liegen, für die ein Zinssatz berechnet werden soll.

First

Mit *First* wird festgelegt, wann die erste Zinsanpassung (d.h. Berechnung eines Zinssatzes) erfolgt. Die Zählung orientiert sich dabei an den in *Interest.Dates* definierten Terminen. Wird für *First* der Wert '1' angegeben, erfolgt die erste Berechnung zum Ende der ersten Zinsperiode, der berechnete Zinssatz gilt dann für die zweite Zinsperiode.

Beispiel: *Für ein Darlehen wurde 'Interest.Dates.First' auf '31.01.2008' und 'Interest.Dates.Interval' auf '1' gesetzt, die erste Zahlung erfolgt also am 31.01.2008, die zweite Zahlung am 28.02.2008. Wird 'Interest.Float.First' auf '1' gesetzt, erfolgt die erste Zinsberechnung für die Periode 31.01.2008 – 28.02.2008. Davor gilt der in 'Interest.NominalRate' hinterlegte Satz.*

AutoFloat

Mit *AutoFloat* kann die Steuerung der Zinsanpassungstermine weitgehend automatisiert werden. Ist *AutoFloat* auf den Wert '*True*' gesetzt, werden nach dem Ausrollen der Zinsanpassungstermine (anhand *First* und *Interval*) diejenigen Anpassungstermine gestrichen, die vor dem Settlement-Datum der Zinskurve (*InterestCurve*) liegen. Dies erfolgt durch Festlegung von *First* auf den ersten Zinsanpassungstermin nach dem Fixing-Datum. Liegt das Settlement-Datum vor Beginn der ersten Zinsperiode, wird im Falle des Wertes '*True*' für *CalculateInitialRate* der Zinssatz der ersten Zinsperiode berechnet. Liegt das Settlement-Datum nach Beginn der ersten Zinsperiode, erfolgt diese Berechnung entsprechend nicht.

Ist *AutoFloat* auf '*False*' gesetzt, erfolgt die Zinsanpassung zum ersten in *Float* definierten Termin. Ist dieser Termin nicht durch die Zinskurve abgedeckt, tritt ein Fehler auf.

Interval

Interval bezeichnet die Anzahl der Zinsperioden, für die ein berechneter Satz jeweils gelten soll, orientiert sich also ebenfalls an den in *Interest.Dates* definierten Terminen.

Beispiel: *Für ein Darlehen wurde 'Interest.Dates.Interval' auf '3' und 'Interest.Float.Interval' auf '2' gesetzt. Die berechneten Zinsen gelten dann jeweils für 6 Monate (eine Zinsperiode dauert 3 Monate, Zinsanpassung alle 2 Zinsperioden)*

Enabled

Um die variable Verzinsung zu aktivieren, setzen Sie *Enabled* auf '*True*'. Wird *Enabled* auf '*False*' gesetzt, wird das Darlehen fest verzinst.

CalculateAmortisation

CalculateAmortisation legt fest, ob bei einem Annuitätendarlehen die Annuität automatisch an einen geänderten Zinssatz angepasst wird ('*True*') oder nicht ('*False*'). Diese Einstellung ignoriert *Amortisation.NoAutomaticRate*, d.h. eine Anpassung der Annuität erfolgt auch dann, wenn *NoAutomaticRate* auf '*True*' gesetzt wurde.

CalculateInitialRate

Wird *First* auf '1' gesetzt, wird der Zinssatz erstmals für die auf die erste Zinszahlung folgende Periode berechnet (d.h. für die 2. Zinsperiode), davor gilt der unter *Interest.NominalRate* angegebene Satz. Soll auch der Satz für die erste Periode berechnet werden, muss *CalculateInitialRate* auf '*True*' gesetzt werden.

7.3.10 ManualAmortisation

Verwendung:	Veränderung einzelner Tilgungsraten
Datentyp:	Collection
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein
Mögliche Werte:	Variablen des Typs baManualAmortisation

Mit der *ManualAmortisation*-Auflistung ist es möglich, einzelne Tilgungsraten (die durch die Eigenschaft *Amortisation* definiert sind) zu manipulieren. Fügen Sie der Auflistung dazu Variablen des Typs *baManualAmortisation* zu, die folgende Elemente beinhalten:

Element	Datentyp	Hinweis
Type	Byte	Gibt an, ob die Tilgungsrate durch einen Wert ersetzt wird oder ob ein Wert addiert wird.
DateNumber	Integer	Laufende Nummer der zu modifizierenden Tilgungszahlung.
Amount	Double	Anpassungsbetrag

Type

Type legt fest, ob eine Tilgungsrate durch den Betrag *Amount* ersetzt werden soll oder ob *Amount* zur Tilgungsrate addiert wird.

Konstante	Wert	Hinweis
baAmortisationAdditional	1	Der Betrag wird zur Tilgungsrate addiert
baAmortisationReplacement	2	Der Betrag ersetzt die Tilgungsrate

DateNumber

Mit *DateNumber* wird festgelegt, auf welche Tilgungsrate sich die Manipulation bezieht. Die Zählung beginnt mit 1 und endet mit *Amortisation.Dates.Count*.

Amount

Amount gibt den Betrag an, durch den die Tilgungsrate ersetzt werden soll oder welcher der Tilgungsrate hinzugefügt werden soll. Tilgungsraten werden mit negativem Vorzeichen geführt (aus der Sicht des Darlehensnehmers).

7.3.11 ManualInterest

Verwendung:	Veränderung einzelner Zinssätze
Datentyp:	Collection
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein
Mögliche Werte:	Variablen des Typs baManualInterest

Mit der *ManualInterest*-Auflistung können einzelne Zinssätze verändert werden. Der Auflistung können Variablen des Typs *baManualInterest* hinzugefügt werden, die folgende Elemente beinhalten:

Element	Datentyp	Hinweis
Type	Byte	Gibt an, ob der Zinssatz durch einen Wert ersetzt wird oder ob ein Wert addiert wird, ob die Zinsmarge aufgeschlagen werden soll und wann keine Veränderung erfolgen soll.
DateNumber	Integer	Laufende Nummer des zu modifizierenden Zinssatzes.
NominalRate	Single	Anpassungszinssatz
Amount	Double	Zinszahlung, aus welcher der Nominalzinssatz berechnet wird.

Type

Type legt fest, ob ein Zinssatz durch den Wert *NominalRate* ersetzt werden soll oder ob *NominalRate* zum Zinssatz addiert wird. *Type* legt weiterhin fest, ob zu *NominalRate* die Zinsmarge des Darlehens addiert werden soll und ob der Zinssatz nur dann verändert werden soll.

Konstante	Wert	Hinweis
baInterestAdditional	1	Der Anpassungszinssatz wird zum Zinssatz der betreffenden Periode addiert.
baInterestReplacement	2	Der Zinssatz der betreffenden Periode wird durch den Anpassungszinssatz ersetzt.
baInterestOnlyNoForwardRate	4	Der Anpassungszinssatz wird nur angewandt, wenn für die Periode kein Forward-Zinssatz ermittelt wurde.
baInterestAddMargin	8	Zum Anpassungszinssatz wird die Zinsmarge des Darlehens addiert.
baInterestCarryOverAllowed	16	Der Übertrag des Zinssatzes in nachfolgende Zinsperioden ist zulässig.
baInterestCalculateRate	32	Anstatt die Höhe der Zinszahlung aus dem in *NominalRate* angegebenen Zinssatz zu berechnen, wird der Nominalzinssatz aus der in *Amount* angegebenen Zinszahlung berechnet. Ein in *NominalRate* übergebener Wert wird ignoriert.

Die einzelnen Konstanten können durch Addition kombiniert werden. Nicht kombiniert werden kann *baInterestAdditional* mit *baInterestReplacement* sowie *baInterestAdditional* mit *baInterestAddMargin* (da die Zinsmarge im Zinssatz der betreffenden Periode bereits enthalten ist). Die Option baInterestCalculateRate kann nicht mit *baInterestAdditional oder baInterestAddMargin kombiniert werden.*

Die Option *baInterestOnlyNoForwardRate* ermöglicht, für variabel verzinsliche Darlehen, für die bereits Zinsperioden fixiert wurden, historische Barwertberechnungen vorzunehmen. Dazu wird zunächst das Darlehen so definiert, dass die Zinssätze für alle Perioden aus einer Zinskurve abgeleitet werden können. Danach werden in *ManualInterest* die bereits fixierten Zinssätze mit den Optionen *baInterestReplacement*, *baInterestOnlyNoForwardRate* und *baInterestAddMargin* eingetragen und die Einstellung *Interest.Float.AutoFloat* auf *True* gesetzt. Das Darlehen kann nun mit historischen Zinskurven berechnet werden, wobei jeweils nur

die manuellen Zinsanpassungen berücksichtigt werden, die zum Zeitpunkt des Settlements der jeweiligen Zinskurve bereits fixiert waren.

Wurde für eine Zinsperiode der Zinssatz verändert, muss der Zinssatz der nachfolgenden Periode entweder ebenfalls manuell gesetzt werden oder er muss als variabler Zinssatz aus der Zinskurve abgeleitet werden können. Ist dies nicht der Fall, wird ein Laufzeitfehler ausgelöst. Durch Addition der Option *baInterestCarryOverAllowed* kann ein manuell gesetzter Zinssatz zum Übertrag in die nachfolgende Periode freigeschaltet werden. Diese Funktion ist mit Vorsicht anzuwenden, da die Gefahr besteht, dass Zinssätze auf Perioden bezogen werden, für die der jeweilige Zinssatz nicht geeignet ist.

DateNumber

Mit *DateNumber* wird festgelegt, auf welchen Zinssatz sich die Manipulation bezieht. Die Zählung beginnt mit 1 und endet mit Interest.*Dates.Count*.

7.3.12 Output

Verwendung:	Rückgabe der Berechnungsergebnisse
Datentyp:	BAOutput
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Nach erfolgreichem Aufruf der Methode *Calculate* verweist Output auf ein Objekt der Klasse *BAOutput*, das alle Berechnungsergebnisse enthält. Das Element *DataArray* enthält folgende Daten:

Konstante	Wert	Hinweis
baHeadDate	1	Datum
baHeadEvent	2	Ereignis
baHeadDrawing	3	Ziehungsbetrag

Konstante	Wert	Hinweis
baHeadInterestDays	4	Zinstage
baHeadInterestCalculation	5	Berechneter Zinsaufwand
baHeadInterestPayment	6	Angesetzte Zinszahlung
baHeadAccruedInterest	7	Saldo des berechneten, aber noch nicht gezahlten Zinsaufwandes
baHeadAmortisationCalculation	8	Berechnete Tilgungsleistung
baHeadAmortisationPayment	9	Angesetzte Tilgungszahlung
baHeadAccruedAmortisation	10	Saldo der berechneten, aber noch nicht gezahlten Tilgungsleistung
baHeadBalance	11	Kreditsaldo
baHeadCashflow	12	Cashflow
baHeadMarginCalculation	13	Berechnete Marge (Teil des Zinsaufwandes)
baHeadMarginPayment	14	Angesetzte Margenzahlung (Teil der Zinszahlung)
baHeadAccruedMargin	15	Saldo der berechneten, aber noch nicht gezahlten Marge
baHeadInterestRate	16	Nominalzinssatz
baHeadDiscountFactor	20	Abzinsungs-/Diskontfaktor
baHeadPresentValue	21	Barwert

7.3.13 Remainder

Verwendung:	Angabe des Restsaldos
Datentyp:	Double
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein
Mögliche Werte:	Zwischen '0' und 'Amount'

7.3.14 Tag

Verwendung:	Frei verwendbares Merkmal
Datentyp:	Variant
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

7.4 Methoden

7.4.1 Calculate

Funktion:	Berechnung des Darlehens
Parameter:	Keiner
Rückgabewert:	Keiner

Calculate berechnet anhand der gewählten Parameter das Darlehen und gibt die Berechnungsergebnisse über die *Output*-Eigenschaft als Verweis auf ein *BAOutput*-Objekt zurück. Nach erfolgreichem Aufruf liefert die Eigenschaft *Calculated* den Wert '*True*' zurück.

7.4.2 Clone

Funktion:	Erstellung einer identischen Kopie des Objekts
Parameter:	Keiner
Rückgabewert:	Objekt des Typs BALoan

Clone erstellt eine 100ige Kopie des Objekts und liefert einen Verweis auf diese Kopie zurück. Modifikationen an der Kopie wirken sich nicht auf das ursprüngliche Objekt aus.

7.4.3 GetAccruedInterest

Funktion:	Ermittlung der aufgelaufenen Zinsen/Stückzinsen zu einem Stichtag
Parameter:	LookupDate (Date)
Rückgabewert:	Stückzinsen (Double)

Zurückgegeben werden die Zinsen, die bis zum Datum *LookupDate* aufgelaufen sind und zur Zahlung fällig wären, wenn das Darlehen zu diesem Stichtag beendet werden würde. In Abzug gebracht werden Zinszahlungen, die planmäßig zum Stichtag fällig sind.

Beispiel: *Am 15.01.2008 ist eine Zinszahlung in Höhe von € 100 angesetzt. 'GetAccruedInterest' gibt für den 15.01.2008 keine Stückzinsen zurück.*

7.5 Ereignisse

7.5.1 AfterCalculation

Auslösung:	Nach erfolgreichem Abschluss der Methode Calculate
Parameter:	Keiner

7.5.2 PropertyChanged

Auslösung:	Nach Veränderung einer berechnungsrelevanten Eigenschaft
Parameter:	Keiner

8 Klasse BAInterestSwap

8.1 Aufgaben & Anwendung

Die Klasse *BAInterestSwap* dient der Abbildung und Bewertung von Zinsswaps. Abgeleitet aus der Bewertung kann der faire Festsatz und der faire Spread ermittelt werden. Da die Definition des Fixed Legs und des Floating Legs über Objekte der Klasse *BALoan* erfolgt, können vielfältige Zahlungsprofile abgebildet werden, nicht nur Plain Vanilla Swaps.

8.2 Eigenschaften

8.2.1 AccruedInterest

Verwendung:	Rückgabe der Stückzinsen zum Bewertungsstichtag
Datentyp:	Double
Lese-/Schreibzugriff:	Ja/Nein

Nach erfolgreichem Aufruf der Methode *Price* gibt *AccruedInterest* die Stückzinsen des Zinsswaps zum Bewertungsstichtag zurück. Die Stückzinsen werden berechnet als Differenz aus Stückzinsen des Floating Legs und Stückzinsen des Fixed Legs.

8.2.2 AutoFixedLeg

Verwendung:	Einstellungen zur Ableitung des Fixed Leg aus dem Floating Leg
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Ist das Zahlungsprofil des Fixed Leg hinsichtlich Terminen, Bezugsbeträgen und der Usancen identisch mit dem Floating Leg oder ähnlich dem Floating Leg, kann das Fixed Leg automatisch aus dem Floating Leg abgeleitet werden. Folgende Einstellungen stehen zur Verfügung:

Konstante	Wert	Hinweis
baNone	0	Keine automatische Erstellung des Fixed Leg
baAutoFixedLeg_Clone	64	Das Fixed Leg wird 1:1 aus dem Floating Leg abgeleitet.
baAutoFixedLeg_Annual	128	Das Fixed Leg wird aus dem Floating Leg abgeleitet, die Zahlungstermine werden jedoch jährlich angesetzt.

Bei *baAutoFixedLeg_Annual* wird die erste Zinszahlung 1 Jahr nach Ziehung angesetzt und danach im Jahresintervall fortgesetzt.

Um die Day Count Convention des Fixed Leg gegenüber der des Floating Leg abzuändern, addieren Sie zu *AutoFixedLeg* den Wert der gewünschten Day Count Convention. Informationen zu den Day Count Conventions finden Sie im Kapitel „3.3.5 GetDayCountFraction“ auf Seite 36.

8.2.3 Calculated

Verwendung:	Rückmeldung, ob der Zinsswap bewertet wurde
Datentyp:	Boolean
Lese-/Schreibzugriff:	Ja/Nein

Calculated liefert den Wert '*True*' zurück, wenn der letzte Aufruf der Methode *Price* erfolgreich war und seitdem keine berechnungsrelevanten Eigenschaften verändert wurden. Da die Methoden *GetFairFixedRate* und *GetFairSpread* auf *Price* basieren, liefert *Calculated* auch nach deren erfolgreichem Aufruf '*True*'. Spiegeln die Bewertungskennzahlen (*CleanPrice, DirtyPrice, AccruedInterest, PVFloating, PVFixed*) nicht die Eigenschaften wider, liefert *Calculated* '*False*' zurück.

8.2.4 CleanPrice

Verwendung:	Berechneter Clean Price (ohne Stückzinsen)
Datentyp:	Double
Lese-/Schreibzugriff:	Ja/Nein

Nach erfolgreichem Aufruf der Methode *Price* enthält *CleanPrice* den Wert des Zinsswaps exklusive Stückzinsen.

Der Berechnungsnachweis für das Fixed Leg ist in *PVFixed* abgespeichert, der Nachweis für das Floating Leg in *PVFloating*.

8.2.5 Description

Verwendung:	Erklärende Bezeichnung des Objekts
Datentyp:	String
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

8.2.6 DirtyPrice

Verwendung:	Berechneter Dirty Price (mit Stückzinsen)
Datentyp:	Double
Lese-/Schreibzugriff:	Ja/Nein

Nach erfolgreichem Aufruf der Methode *Price* enthält *DirtyPrice* den Wert des Zinsswaps inklusive Stückzinsen.

Der Berechnungsnachweis für das Fixed Leg ist in *PVFixed* abgespeichert, der Nachweis für das Floating Leg in *PVFloating*.

8.2.7 FixedRate

Verwendung:	Festsatz des Zinsswaps
Datentyp:	Single
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

FixedRate gibt den Festsatz des Zinsswaps an, also den Nominalzinssatz des Fixed Leg. Der Wert muss in der Form „P/100“ vorliegen (4% = 4/100 = 0.04). Soll das Fixed Leg automatisch aus dem Floating Leg abgeleitet werden (*AutoFixedLeg* > 0), wird für das Fixed Leg der in FixedRate übergebene Zinssatz übernommen. Wird das Fixed Leg explizit angegeben (*AutoFixedLeg* = 0), spielt ein in *FixedRate* gespeicherter, vom Nominalzinssatz des Fixed Leg abweichender Zinssatz keine Rolle. Nach erfolgreichem Aufruf der Methode *Price* enthält *FixedRate* den für den Zinsswap verwendeten Festsatz.

Um den fairen Festsatz zu bestimmen, verwenden Sie die Methode *GetFairFixedRate*.

8.2.8 InterestCurve_Payer

Verwendung:	Zinskurve des Festzinszahlers
Datentyp:	BAInterestCurve
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Anhand der in *InterestCurve_Payer* beschriebenen Zinskurve werden die Zahlungen des Festzinszahlers (d.h. die Zinszahlungen des Fixed Leg) abgezinst. Die Fixing-Termine und die Settlement-Termine von *InterestCurve_Payer* und *InterestCurve_Receiver* müssen jeweils übereinstimmen.

8.2.9 InterestCurve_Receiver

Verwendung:	Zinskurve des Festzinsempfängers
Datentyp:	BAInterestCurve
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Anhand der in *InterestCurve_Receiver* beschriebenen Zinskurve werden die Zahlungen des Festzinsempfängers (d.h. die Zinszahlungen des Floating Leg) abgezinst. Die Fixing-Termine und die Settlement-Termine von *InterestCurve_Payer* und *InterestCurve_Receiver* müssen jeweils übereinstimmen.

8.2.10 PVFixed

Verwendung:	Berechnungsnachweis Fixed Leg
Datentyp:	baPresentValue
Lese-/Schreibzugriff:	Ja/Nein

Nach der Bewertung des Zinsswaps enthält *PVFixed* einen Berechnungsnachweis des Barwerts des Fixed Leg. Folgende Informationen sind im Element *Data* aufgeführt:

- Datum (*baHeadDate*)
- Diskontfaktor (*baHeadDiscountFactor*)
- Zinszahlung (*baHeadInterestPayment*)
- Barwert der Zinszahlung (*baHeadPresentValue*)

8.2.11 PVFloating

Verwendung:	Berechnungsnachweis Floating Leg
Datentyp:	baPresentValue
Lese-/Schreibzugriff:	Ja/Nein

Nach der Bewertung des Zinsswaps enthält *PVFloating* einen Berechnungsnachweis des Barwerts des Floating Leg. Es sind die gleichen Informationen aufgeführt wie in *PVFixed*.

8.2.12 Spread

Verwendung:	Spread auf das Floating Leg des Zinsswaps
Datentyp:	Single
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Spread gibt den Aufschlag auf den variablen Zinssatz des Floating Legs an. Der Wert muss in der Form „P/100“ vorliegen (4% = 4/100 = 0.04). Er wird jeweils zum Forward-Satz addiert, der sich aus der Zinskurve ergibt, auf der die variable Verzinsung von *LoanFloating* basiert.

Um den fairen Spread zu bestimmen, verwenden Sie die Methode *GetFairSpread*.

8.2.13 SwapType

Verwendung:	Definition des Zinsswap-Typs
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Mit *SwapType* wird festgelegt, ob es sich um einen Payer- oder einen Receiver-Zinsswap handelt. Der Unterschied zwischen den beiden Typen besteht im Vorzeichen der ermittelten Bewertungsergebnisse:

- Bei der Bewertung eines Payer-Zinsswaps werden die Ergebnisse aus Sicht des Festzins-Zahlers dargestellt, d.h. der Fair Value ist negativ, wenn das Zinsniveau unterhalb des Festsatzes (*FixedRate*) liegt und positiv, wenn das Zinsniveau oberhalb des Festsatzes liegt.

- Bei der Bewertung eines Receiver-Zinsswaps entspricht die Darstellung der Ergebnisse entsprechend spiegelbildlich der Sicht des Zahlers der variablen Zinsen. Liegt das Zinsniveau unterhalb des Festsatzes, ist der Fair Value positiv. Umgekehrt ist der Fair Value bei einem Zinsniveau oberhalb des Festsatzes negativ.

Konstante	Wert	Bedeutung
baInterestSwapType_Payer	1	Tausch variabler gegen festen Zinssatz, Bewertung aus Sicht des Festzins-Zahlers
baInterestSwapType_Receiver	2	Tausch fester gegen variablen Zinssatz, Bewertung aus Sicht des Festzins-Empfängers

8.2.14 Tag

Verwendung:	Frei verwendbares Merkmal
Datentyp:	Variant
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

8.2.15 ValuationMode

Verwendung:	Definition des Bewertungsmodus des Zinsswaps
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

ValuationMode steuert, ob die Bewertung des Zinsswaps anhand von Brutto- oder Netto-Zahlungsströmen erfolgt. Dies ist nur dann relevant, wenn die Payer- und Receiver-Zinskurven voneinander abweichen, z.B. weil sie unterschiedlichen Adressenausfallrisiken unterliegen.

- Die Brutto-Bewertung zinst alle festen Zinszahlungen mit der Payer-Zinskurve und alle variablen Zinszahlungen mit der Receiver-Zinskurve ab. Die sich daraus ergebenden Kapitalwerte werden saldiert und ergeben den Fair Value des Zinsswaps.

- Die Netto-Bewertung saldiert zu jedem Stichtag die festen Zinszahlungen mit den variablen Zinszahlungen. Ergibt sich ein Saldo zu Lasten des Festzins-Zahlers, wird er mit der Payer-Zinskurve abgezinst. Ergibt sich ein Saldo zu Lasten des Festzins-Empfängers, wird er mit der Receiver-Zinskurve abgezinst.

Konstante	Wert	Bedeutung
baInterestSwapValuationModeGross	0	Bewertung anhand von Brutto-Zahlungsströmen
baInterestSwapValuationModeNet	1	Bewertung anhand von Netto-Zahlungsströmen

8.3 Methoden

8.3.1 GetFairFixedRate

Funktion:	Ermittlung des fairen Festsatzes des Zinsswaps
Parameter:	Keiner
Rückgabewert:	Keiner

GetFairFixedRate ermittelt denjenigen Festsatz, der zu einem Fair Value des Zinsswaps von Null führt. Nach erfolgreichem Aufruf enthält *FixedRate* den ermittelten Wert.

8.3.2 GetFairSpread

Funktion:	Ermittlung des fairen Spreads des Zinsswaps
Parameter:	Keiner
Rückgabewert:	Keiner

GetFairSpread ermittelt denjenigen Spread auf das Floating Leg, der zu einem Fair Value des Zinsswaps von Null führt. Nach erfolgreichem Aufruf enthält *Spread* den ermittelten Wert.

8.3.3 Price

Funktion:	Ermittlung des Fair Value des Zinsswaps
Parameter:	Keiner
Rückgabewert:	Keiner

Price berechnet den Dirty Price (Fair Value, incl. Stückzinsen), die Stückzinsen (AccruedInterest) sowie den Clean Price (Dirty Price ./. Stückzinsen) des Zinsswaps.

Formeln & Berechnungsgrundlagen

Berechnung Clean Price bei Brutto-Bewertung (Sicht: Payer)

$$Clean\ Price = \sum_{a=1}^{t} Var.\ Zinsen_a * df_{Receiver,a} - \sum_{a=1}^{t} Feste\ Zinsen_a * df_{Payer,a}$$

Im Falle der Bewertung aus Sicht des Festzinsempfängers (Receiver) entspricht der Clean Price dem Ergebnis der o.g. Formel mit umgekehrtem Vorzeichen.

Berechnung Clean Price bei Netto-Bewertung (Sicht: Payer)

$$Clean\ Price = \sum_{a=1}^{t} Max(Var.\ Zinsen_a - Feste\ Zinsen_a; 0) * df_{Receiver,a} - \sum_{a=1}^{t} Max(Feste\ Zinsen_a - Var.Zinsen_a; 0) * df_{Payer,a}$$

Berechnung Clean Price bei Netto-Bewertung (Sicht: Receiver)

$$Clean\ Price = \sum_{a=1}^{t} Max(Feste\ Zinsen_a - Var.Zinsen_a; 0) * df_{Payer,a}$$

$$- \sum_{a=1}^{t} Max(Var.\ Zinsen_a - Feste\ Zinsen_a; 0) * df_{Receiver,a}$$

8.4 Ereignisse

8.4.1 AfterCalculation

Auslösung:	Nach erfolgreichem Abschluss der Methode Price
Parameter:	Keiner

8.4.2 PropertyChanged

Auslösung:	Nach Veränderung einer berechnungsrelevanten Eigenschaft
Parameter:	Keiner

9 Klasse BAFXForward

9.1 Aufgaben & Anwendung

Mit der Klasse *BAFXForward* können Devisentermingeschäfte (Solo-Termingeschäfte/Outrights) bewertet werden.

9.2 Eigenschaften

9.2.1 Calculated

Verwendung:	Rückmeldung, ob die Geschäfte bewertet wurden
Datentyp:	Boolean
Lese-/Schreibzugriff:	Ja/Nein

Calculated liefert den Wert '*True*' zurück, wenn der letzte Aufruf der Methode *Price* erfolgreich war und seitdem keine berechnungsrelevanten Eigenschaften verändert wurden. Spiegelt *Result* nicht die Eigenschaften wider, liefert *Calculated* '*False*' zurück.

9.2.2 Contracts

Verwendung:	Definition der Devisentermingeschäfte
Datentyp:	Collection
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Werte des Typs baFXForwardContract

In der *Contracts*-Auflistung werden die zu bewertenden Devisentermingeschäfte definiert. Jedes Geschäft wird mit einer Variablen des Typs *baFXForwardContract* beschrieben, der folgende Elemente aufweist:

Element	Datentyp	Hinweis
FixingDate	Date	Abschlussdatum
CurrDomestic	String	Heimwährung
CurrForeign	String	Fremdwährung
AmountDomestic	Double	Betrag in Heimwährung
AmountForeign	Double	Betrag in Fremdwährung
FXRateFixed	Single	Fixierter Wechselkurs
ValueDate	Date	Valuadatum
Reference	String	Kennung/Referenz
Counterparty	String	Handelspartner

FixingDate & ValueDate

Das Fixing Date bezeichnet das Datum, an dem das Geschäft abgeschlossen wurde. Das Value Date ist der zukünftige Termin, zu dem das Geschäft durch Zahlung und Entgegennahme der vereinbarten Beträge erfüllt wird. Das Value Date muss sowohl durch die Zinskurve als auch die Devisenterminkurve abgedeckt sein (d.h. es muss vor der Eigenschaft LastDate der beiden Kurven liegen).

CurrDomestic & CurrForeign

CurrDomestic (Heimwährung) und *CurrForeign* (Fremdwährung) bezeichnen das gehandelte Währungspaar. Alle Geschäfte in der *Contracts*-Auflistung müssen dasselbe Währungspaar aufweisen, das zudem mit dem Währungspaar der Devisenterminkurve (*FXForwardCurve*) übereinstimmen muss. *CurrDomestic* muss mit der Währung der Zinskurve (*InterestCurve*) übereinstimmen, da die Geschäfte in der Heimwährung bewertet werden.

AmountDomestic, AmountForeign & FXRateFixed

AmountDomestic bezeichnet den Betrag in Heimwährung, der zum Value Date gezahlt oder entgegengenommen werden muss. *AmountForeign* bezeichnet den entsprechenden Betrag in Fremdwährung. Ein positives Vorzeichen gibt an, dass der Betrag entgegengenommen wird, ein negatives Vorzeichen gibt an, dass der Betrag gezahlt werden muss. Die beiden Beträge müssen

unterschiedliche Vorzeichen aufweisen. Ist der fixierte Wechselkurs *FXRateFixed* nicht angegeben, wird er aus *AmountDomestic* und *AmountForeign* bestimmt. Ist hingegen *FXRateFixed* angegeben, wird *AmountForeign* berechnet, wenn *AmountDomestic* angegeben ist, umgekehrt wird *AmountDomestic* berechnet, wenn *AmountForeign* angegeben ist.

Reference & Counterparty

Diese beiden Eigenschaften dienen der Identifizierung eines Geschäfts, indem Handelspartner und Geschäftsnummer oder eine anderes Merkmal gespeichert werden können.

9.2.3 Description

Verwendung:	Erklärende Bezeichnung des Objekts
Datentyp:	String
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

9.2.4 ErrorContract

Verwendung:	Kennzeichnung eines fehlerhaften Geschäfts
Datentyp:	Integer
Lese-/Schreibzugriff:	Ja/Nein

Tritt beim Aufruf von *Price* ein Laufzeitfehler auf, der einem fehlerhaften Geschäft in der Contracts-Auflistung geschuldet ist (unvollständige Angaben, falsches Währungspaar o.ä.), wird in *ErrorContract* die Nummer des betroffenen Geschäfts hinterlegt. Die Zählung beginnt mit '1'.

9.2.5 FXForwardCurve

Verwendung:	Devisenterminkurve zur Ermittlung der Terminkurse
Datentyp:	BAFXForwardCurve
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

FXForwardCurve muss auf eine Devisenterminkurve verweisen, deren Währungspaar mit dem der Geschäfte der *Contracts*-Auflistung übereinstimmt. Weiterhin muss das Settlement-Datum der Devisenterminkurve mit dem Settlement-Datum der Zinskurve (*InterestCurve*) übereinstimmen. Die Devisenterminkurve muss die Laufzeit der Geschäfte vollständig abdecken, das früheste Valutadatum darf also nicht vor dem Settlement-Datum der Devisenterminkurve liegen, das späteste nicht nach *LastDate*.

9.2.6 InterestCurve

Verwendung:	Zinskurve zur Abzinsung der Zahlungen aus den Geschäften
Datentyp:	BAFXInterestCurve
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Die Berechnung der Barwerte der Geschäfte erfolgt anhand der Zinskurve *InterestCurve*. Die Währung der Zinskurve muss mit der Heimwährung der Devisentermingeschäfte übereinstimmen. Hinsichtlich dem Settlement-Datum und der durch die Zinskurve abzudeckenden Zeitspanne gelten die Ausführungen im Abschnitt „9.2.5 “ entsprechend.

9.2.7 Result

Verwendung:	Bewertungsergebnis
Datentyp:	BAOutput
Lese-/Schreibzugriff:	Ja/Nein

Nach erfolgreichem Aufruf von *Price* enthält *Result* die Bewertung der in der *Contracts*-Auflistung definierten Geschäfte.

9.2.8 Tag

Verwendung:	Frei verwendbares Merkmal
Datentyp:	Variant
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

9.2.9 ValuationMode

Verwendung:	Art der Ermittlung der Bewertungs-Terminkurse
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Mit *ValuationMode* kann eingestellt werden, wie zum Bewertungsstichtag der Devisenterminkurs ermittelt wird. Folgende Modi stehen zur Verfügung:

Konstante	Wert	Kursart Devisenterminkurs
baValuationModeNewContract	0	Gleiche Kursart wie Devisentermingeschäft
baValuationModeCountertrade	1	Entgegengesetzte Kursart zum Devisentermingeschäft
baValuationModeMid	2	Mittelkurse

New Contract

Bei dieser Methode wird die Kursart so gewählt, dass sie mit der Kursart des zu bewertenden Geschäfts übereinstimmt. Wird im Geschäft Fremdwährung gegen Heimwährung verkauft, wird der Briefkurs (Ask) herangezogen. Wird im Geschäft Fremdwährung gegen Heimwährung gekauft,

wird der Geldkurs (Bid) verwendet. Die Bewertung erfolgt somit aus dem Blickwinkel, zu welchem Kurs das Geschäft zum Bewertungsstichtag abgeschlossen werden könnte.

Countertrade

Hier wird die Kursart entgegengesetzt zur Kursart des zu bewertenden Geschäfts gewählt. Wird im Geschäft Fremdwährung gegen Heimwährung verkauft, wird der Geldkurs (Bid) herangezogen und umgekehrt. Die Bewertung beantwortet die Frage, zu welchem Kurs das Geschäft zum Bewertungsstichtag glattgestellt werden könnte.

Mittelkurs

Unabhängig von der Kursart des jeweiligen Geschäfts wird für die Bewertung stets der Mittelkurs herangezogen.

9.3 Methoden

9.3.1 Price

Funktion:	Ermittlung des Fair Value der Devisentermingeschäfte
Parameter:	Keiner
Rückgabewert:	Keiner

Price bewertet die in der Contracts-Auflistung enthaltenen Geschäfte und speichert das Ergebnis in *Result*. Folgende Daten werden ausgegeben:

- Alle Daten, die in *baFXForwardContract* enthalten sind (die Definition der zu bewertenden Geschäfte)

- **Kennzeichnung des Geschäfts** *(baHeadCurrCode)*
  Kennzeichnung des Geschäfts als Verkettung der Währungscodes *CurrDomestic* und *CurrForeign*. Zuerst wird die gekaufte Währung aufgeführt, gefolgt von der verkauften Währung (Bsp. „USDEUR“, wenn USD gegen EUR gekauft werden).

- **Bewertungskurs** *(baHeadFXRateValuationDate)*

  Devisenterminkurs, der sich aus der Devisenterminkurve *BAFXForwardCurve* für das Valutadatum ergibt und der zur Bewertung des Geschäfts herangezogen wird.

- **Kurstyp des Bewertungskurses** (*baHeadFXRateType*)

  Kennzeichnung, ob es sich beim Bewertungskurs um den Geld-, Brief- oder Mittelkurs handelt.

- **Gewinn/Verlust** (*baHeadProfitLoss*)

  Gewinn oder Verlust aus dem Geschäft. Dieser Betrag wird stets in der Heimwährung berechnet und ermittelt sich als Differenz aus dem zu zahlenden oder zu empfangenden Betrag in Heimwährung und dem zu zahlenden oder zu empfangenden Betrag in Fremdwährung, der mit dem Bewertungskurs in die Heimwährung umgerechnet wird:

  Gewinn/Verlust = Betrag in HW ./. (Betrag in FW / Bewertungskurs)

- **Diskontfaktor** (*baHeadDiscountFactor*)

  Abzinsungsfaktor, der sich aus der Zinskurve InterestCurve für das Valutadatum ergibt.

- **Barwert** (*baHeadPresentValue*)

  Barwertiger Gewinn/Verlust

  Barwert = Gewinn/Verlust * Diskontfaktor

9.4 Ereignisse

9.4.1 AfterCalculation

Auslösung:	Nach erfolgreichem Abschluss der Methode Price
Parameter:	Keiner

9.4.2 PropertyChanged

Auslösung:	Nach Veränderung einer berechnungsrelevanten Eigenschaft
Parameter:	Keiner

10 Klasse BAPrepaymentCompensation

10.1 Aufgaben & Anwendung

Mit Objekten der Klasse *BAPrepaymentCompensation* kann für vorfällig abzulösende Darlehen eine Vorfälligkeitsentschädigung berechnet werden. In die Berechnung können verschiedene Komponenten (ersparte Risiko- und Administrationskosten, Bearbeitungskosten) einbezogen werden. Weiterhin kann die Vorfälligkeitsentschädigung in den Zinsverschlechterungsschaden (Änderung des Zinsniveaus gegenüber dem Zeitpunkt der Zinsfixierung des Darlehens) und den Zinsmargenschaden (entgangene Marge durch vorzeitige Beendigung) unterteilt werden.

10.2 Eigenschaften

10.2.1 Description

Verwendung:	Erklärende Bezeichnung des Objekts
Datentyp:	String
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

10.2.2 Compensation

Verwendung:	Berechnete Vorfälligkeitsentschädigung
Datentyp:	Double
Lese-/Schreibzugriff:	Ja/Nein

Nach erfolgreichem Aufruf von *Calculate* enthält *Compensation* die berechnete Vorfälligkeitsentschädigung in Geldeinheiten. Diese wird wie folgt ermittelt:

Barwert der ausstehenden Zahlungen (*PVCashflow*)
./. Aktueller Saldo (*CurrentBalance*)
\+ Bearbeitungskosten (*PVCostOfHandling*)

./. Ersparte Risikokosten (*PVCostOfRisk*)

./. Ersparte Administrationskosten (*PVCostOfAdministration*)

Weist *Compensation* ein negatives Vorzeichen auf, so steht der Betrag dem Kreditgeber zu, andernfalls dem Kreditnehmer.

Der Zinsmargenschaden ist in der Differenz aus dem Barwert der ausstehenden Zahlungen und dem aktuellen Saldo enthalten. Ein separater Ausweis ist mit *PVLostMargin* möglich.

10.2.3 CostOfAdministration

Verwendung:	Ersparte Administrationskosten des Darlehensgebers
Datentyp:	Double
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Aufgrund der Beendigung des Darlehens entfallen für den Darlehensgeber Verwaltungskosten, die sonst bis zum Ende des Darlehensverhältnisses angefallen wären. *CostOfAdministration* bezeichnet die ersparten Verwaltungskosten in Geldeinheiten pro Jahr. Sie werden bis zum Ende des Darlehens bzw. bis *FixedUntil* angesetzt und barwertig in die Schadensberechnung einbezogen. *CostOfAdministration* sind mit positivem Vorzeichen anzugeben, da sie aus Sicht des Darlehensnehmers zufließen.

Der Berechnungsnachweis ist nach erfolgreichem Aufruf von *Calculate* in *PVCostOfAdministration* enthalten.

10.2.4 CostOfHandling

Verwendung:	Bearbeitungskosten des Darlehensgebers für die vorzeitige Beendigung
Datentyp:	Double
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Durch die vorzeitige Beendigung können dem Darlehensgeber einmalig Bearbeitungskosten entstehen, die er in der Vorfälligkeitsentschädigung berücksichtigen kann. Mit *CostOfHandling* können diese Bearbeitungskosten in Geldeinheiten angegeben werden. Dieser Betrag ist mit negativem Vorzeichen anzugeben (da er aus Sicht des Darlehensnehmers abfließt).

10.2.5 CostOfRisk

Verwendung:	Ersparte Risikokosten des Darlehensgebers
Datentyp:	Single
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Analog zu den Verwaltungskosten entfallen für den Darlehensgeber Risikokosten, wenn das Darlehen vorzeitig zurückgeführt wird. Diese Risikokosten bemessen sich prozentual nach dem jeweils ausstehenden Kapital. Mit *CostOfRisk* kann die Höhe der ersparten Risikokosten p.a. in der Form „P/100“ angegeben werden (4% = 4/100 = 0.04). Die Risikokosten werden bis zum Ende des Darlehens bzw. bis FixedUntil angesetzt und barwertig in die Schadensberechnung einbezogen.

Der Berechnungsnachweis ist nach erfolgreichem Aufruf von *Calculate* in *PVCostOfRisk* enthalten.

10.2.6 CurrentBalance

Verwendung:	Saldo zum Zeitpunkt der vorzeitigen Ablösung
Datentyp:	Double
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein (Ja, wenn LoanNoRefresh = 'True')

CurrentBalance bezeichnet den Betrag, der bei Ablösung des Darlehens zum Settlement-Datum der Zinskurve (*InterestCurve*) an den Darlehensgeber zu entrichten wäre, wenn keine Vorfälligkeitsentschädigung anfallen würde. *CurrentBalance* wird berechnet aus dem ausstehenden Kreditsaldo und den auf das Ablösedatum abgegrenzten Zinsen. Es wird mit positivem Vorzeichen geführt.

Sofern *LoanNoRefresh* auf '*True*' gestellt ist, muss *CurrentBalance* vor Aufruf von *Calculate* angegeben werden, andernfalls enthält *CurrentBalance* nach Aufruf von *Calculate* den berechneten Wert.

10.2.7 FixedUntil

Verwendung:	Ende der Zinsbindung des Darlehens
Datentyp:	Date
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

FixedUntil bezeichnet ein Datum vor dem regulären Ablauf des Darlehens, an dem die Zinsbindung endet. Für die Berechnung der Vorfälligkeitsentschädigung werden nur solche Zahlungen einbezogen, die bis zum Ende der Zinsbindung erfolgen. Zum Ende der Zinsbindung setzte *Calculate* eine vollständige Rückführung des Darlehens an.

10.2.8 InterestCurve

Verwendung:	Zinskurve zur Abzinsung der ausstehenden Zahlungsströme
Datentyp:	BAInterestCurve
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Mit den aus der in *InterestCurve* beschriebenen Zinskurve abgeleiteten Diskontfaktoren werden die planmäßig ausstehenden Zahlungsströme des abzulösenden Darlehens *Loan* bis zum Ende der Zinsbindung (*FixedUntil*) abgezinst. Das Darlehen wird zum Settlement-Datum der Zinskurve abgelöst.

10.2.9 Loan

Verwendung:	Beschreibung des abzulösenden Darlehens
Datentyp:	BALoan
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Loan beschreibt das Darlehen, für das die Vorfälligkeitsentschädigung berechnet werden soll.

10.2.10 LoanNoRefresh

Verwendung:	Unterbindung einer Neuberechnung des Darlehens
Datentyp:	Boolean
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Sofern *LoanNoRefresh* auf '*False*' gesetzt ist, berechnet die Methode *Calculate* aus dem Darlehen automatisch alle Parameter, die zur Ermittlung der Vorfälligkeitsentschädigung erforderlich sind. Soll das Darlehen nicht neu berechnet werden (z.B. weil *Loan.Output* ein

Zahlungsstrom-Profil enthält, dass sich nicht standardmäßig berechnen lässt), muss *LoanNoRefresh* auf '*True*' gesetzt werden. Dabei ist zu beachten:

- Der Saldo zum Ablösezeitpunkt ist in *CurrentBalance* explizit anzugeben, er kann nicht berechnet werden.
- Endet die Zinsbindung des Darlehens vor dem Ende der Laufzeit, so muss dies im Zahlungsstrom-Profil des Darlehens berücksichtigt sein. Ein in *FixedUntil* angegebenes Datum kann nicht berücksichtigt werden, da hierfür das Darlehen neu berechnet werden müsste.
- Eine im Darlehen hinterlegte Zinsmarge (*Margin*) wird als ersparte Risikokosten (*CostOfRisk*) interpretiert.
- Der Zinsmargenschaden kann nicht berechnet werden.

10.2.11 PVCashflow

Verwendung:	Berechnungsnachweis ausstehende Zahlungen
Datentyp:	baPresentValue
Lese-/Schreibzugriff:	Ja/Nein

Nach erfolgreichem Aufruf von *Calculate* enthält *PVCashflow* einen Berechnungsnachweis des Barwerts der ausstehenden Zahlungen bis zum regulären Ablauf des Darlehens bzw. bis zum Ende der Zinsbindung (*FixedUntil*). Folgende Informationen sind im Element *Data* aufgeführt:

- Datum (*baHeadDate*)
- Diskontfaktor (*baHeadDiscountFactor*)
- Cashflow (*baHeadCashflow*)
- Barwert des Cashflows (*baHeadPresentValue*)

10.2.12 PVCostOfAdministration

Verwendung:	Berechnungsnachweis ersparte Verwaltungskosten
Datentyp:	baPresentValue
Lese-/Schreibzugriff:	Ja/Nein

Nach erfolgreichem Aufruf von *Calculate* enthält *PVCashflow* einen Berechnungsnachweis des Barwerts der Verwaltungskosten, die bis zum regulären Ablauf des Darlehens bzw. bis zum Ende der Zinsbindung (*FixedUntil*) entstanden wären. Folgende Informationen sind im Element *Data* aufgeführt:

- Datum (*baHeadDate*)
- Diskontfaktor (*baHeadDiscountFactor*)
- Verwaltungskosten (*baHeadCostOfAdministration*)
- Barwert der Verwaltungskosten (*baHeadPresentValue*)

10.2.13 PVCostOfRisk

Verwendung:	Berechnungsnachweis ersparte Risikokosten
Datentyp:	baPresentValue
Lese-/Schreibzugriff:	Ja/Nein

Nach erfolgreichem Aufruf von *Calculate* enthält *PVCashflow* einen Berechnungsnachweis des Barwerts der Risikokosten, die bis zum regulären Ablauf des Darlehens bzw. bis zum Ende der Zinsbindung (*FixedUntil*) entstanden wären. Folgende Informationen sind im Element *Data* aufgeführt:

- Datum (*baHeadDate*)
- Diskontfaktor (*baHeadDiscountFactor*)
- Risikokosten (*baHeadCostOfRisk*)
- Barwert der Risikokosten (*baHeadPresentValue*)

10.2.14 PVLostMargin

Verwendung:	Berechnungsnachweis entgangene Zinsmarge
Datentyp:	baPresentValue
Lese-/Schreibzugriff:	Ja/Nein

Nach erfolgreichem Aufruf von *Calculate* enthält *PVCashflow* einen Berechnungsnachweis des Barwerts der Zinsmargen, die bis zum regulären Ablauf des Darlehens bzw. bis zum Ende der Zinsbindung (*FixedUntil*) entstanden wären. Folgende Informationen sind im Element *Data* aufgeführt:

- Datum (*baHeadDate*)
- Diskontfaktor (*baHeadDiscountFactor*)
- Zinsmarge (*baHeadMarginPayment*)
- Barwert der Zinsmarge (*baHeadPresentValue*)

10.2.15 Tag

Verwendung:	Frei verwendbares Merkmal
Datentyp:	Variant
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

10.3 Methoden

10.3.1 Calculate

Funktion:	Ermittlung der Vorfälligkeitsentschädigung
Parameter:	Keine
Rückgabewert:	Keiner

10.4 Ereignisse

10.4.1 AfterCalculation

Auslösung:	Nach erfolgreichem Abschluss der Methode Calculate
Parameter:	Keiner

10.4.2 PropertyChanged

Auslösung:	Nach Veränderung einer berechnungsrelevanten Eigenschaft
Parameter:	Keiner

11 Klasse BALoanPricing

11.1 Aufgaben & Anwendung

Die Klasse *BALoanPricing* dient der Ermittlung von fairen Darlehenszinssätzen auf Grundlage einer gegebenen Zinskurve. So kann z.B. ein von einem Kreditinstitut angegebener Zinseinstand nachvollzogen werden. Unterstützt werden zwei verschiedene Berechnungsverfahren: Die Barwert-Methode und die Methode des tilgungsschichten-gewichteten Einstandes („Praktikermethode“).

11.2 Eigenschaften

11.2.1 CalculationMode

Verwendung:	Legt fest, wie der Zinssatz ermittelt wird
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Mit *CalculationMode* wird festgelegt, auf welche Art der Zinssatz für das Darlehen berechnet werden soll. Es stehen zwei Verfahren zur Auswahl:

Konstante	Wert	Hinweis
baInterestPresentValue	0	Barwert-Methode (exaktes Verfahren)
BaInterestWeightedAmortisation	1	Tilgungsschichten-gewichteter Einstand (Praktiker-Verfahren)

- **Barwert-Methode**

  Bei der Barwert-Methode wird der Nominalzinssatz des Darlehens so festgelegt, dass der anhand der Zinskurve *InterestCurve* berechnete Kapitalwert des Darlehens Null ist.

- **Tilgungsschichten-gewichteter Einstand**

  Bei diesem Verfahren wird das Darlehen in so viele endfällige Einzeldarlehen zerlegt, wie Tilgungszahlungen vorliegen. Die Laufzeit eines solchen Einzeldarlehens entspricht dabei der Zeitspanne zwischen Auszahlung des Darlehens und Fälligkeit der Tilgungszahlung, aus der es abgeleitet wurde, der Darlehensbetrag entspricht dem Betrag der Tilgungszahlung. Jedem Einzeldarlehen wird der laufzeitkongruente Zinssatz aus der Zinskurve zugewiesen, um durch Multiplikation mit der Laufzeit und dem Darlehensbetrag die Zinszahlungen des Teildarlehens zu ermitteln. Anschließend wird die Summe aller Zinszahlungen aller Teildarlehen durch die Laufzeiten und die Darlehensbeträge dividiert – das Ergebnis ist der durchschnittliche Jahreszinssatz über alle Teildarlehen. Der so ermittelte Zinssatz ist finanzmathematisch nicht korrekt, das Verfahren eignet sich jedoch recht gut, um ohne großen Aufwand „Schätzwerte“ zu ermitteln.

11.2.2 Description

Verwendung:	Erklärende Bezeichnung des Objekts
Datentyp:	String
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

11.2.3 InterestCurve

Verwendung:	Zinskurve, aus welcher der Zinseinstand abgeleitet wird
Datentyp:	BAInterestCurve
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Ja

Mittels *InterestCurve* werden die Zahlungen des Darlehens abgezinst (bei der Barwert-Methode) bzw. die Zinssätze ermittelt, die in den tilgungsschichten-gewichteten Einstand eingehen.

11.2.4 InterestRate

Verwendung:	Berechneter Zinssatz
Datentyp:	Single
Lese-/Schreibzugriff:	Ja/Nein

Nach erfolgreichem Aufruf von *Calculate* enthält *InterestRate* den ermittelten Nominalzinssatz einschließlich Marge.

11.2.5 IterationChange

Verwendung:	Kleinste zulässige Zinsänderung bei iterativer Zinsermittlung
Datentyp:	Single
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Im Falle einer iterativen Ermittlung des Zinssatzes legt *IterationChange* gemeinsam mit *IterationCount* fest, wie oft die iterative Schleife durchlaufen wird. Die Berechnung endet, sobald

- die Veränderung des Zinssatzes gegenüber dem vorherigen Berechnungsschritt kleiner als *IterationChange* („P/100“: 4% = 4/100 = 0.04) ausfällt **oder**

- die vorgegebene Anzahl an Wiederholungen (*IterationCount*) erreicht ist.

Info: Iterative Einstandsermittlung

Der Nominalzinssatz wird aus den Cashflows (Barwert-Methode) bzw. Tilgungszahlungen (tilgungsschichten-gewichteter Einstand) abgeleitet, deren Struktur wiederum vom Nominalzinssatz abhängt. Es ist daher möglich, dass ein berechneter Zinssatz das Darlehen so verändert, dass eine erneute Berechnung des Zinssatzes auf Grundlage des veränderten Darlehens zu einem anderen Ergebnis kommen würde. Dies ist insbesondere bei Annuitätendarlehen der Fall, da sich deren Tilgungsstruktur abhängig vom Zinssatz verändert. Daher sollte die Berechnung des Zinssatzes so oft wiederholt werden, bis keine (relevante) Änderung mehr auftritt.

11.2.6 IterationCount

Verwendung:	Max. Anzahl der Berechnungsdurchläufe bei iterativer Zinsermittlung
Datentyp:	Byte
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Informationen zu *IterationCount* finden Sie im vorhergehenden Kapitel.

11.2.7 MarginAdjustment

Verwendung:	Berücksichtigung der Zinsmarge bei der Zinsermittlung
Datentyp:	Boolean
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

Der berechnete Zinssatz *InterestRate* wird stets einschließlich einer ggf. im Darlehen *Loan* definierten Zinsmarge zurückgegeben. Mit *MarginAdjustment* kann gesteuert werden, ob die Zinsmarge bei der Ermittlung des Einstandssatzes berücksichtigt werden soll.

- Wird *MarginAdjustment* auf '*False*' gesetzt, bleibt die Zinsmarge bei der Ermittlung des Zinseinstandes außer Ansatz. Die Marge wird nicht in das Darlehen einbezogen, aus dem der

Einstandssatz abgeleitet wird und erst nach Ermittlung des Einstandssatzes diesem aufgeschlagen.

- Wird *MarginAdjustment* auf '*True*' gesetzt, wird die Zinsmarge in die Ermittlung des Zinseinstandes einbezogen, indem die Marge im Darlehen, aus dem der Einstandssatz abgeleitet wird, berücksichtigt wird. Die Barwert-Methode ermittelt in diesem Fall nicht den Zinssatz, der zu einem Kapitalwert von Null führt, sondern wählt diesen so, dass Kapitalwert und Margenbarwert identisch sind.

11.2.8 PresentValue

Verwendung:	Barwerte der Cashflows des Darlehens als Berechnungsnachweis
Datentyp:	baPresentValue
Lese-/Schreibzugriff:	Ja/Nein

Nach erfolgreichem Aufruf von *Calculate* enthält *PresentValue* die Barwerte der Cashflows des anhand des ermittelten Zinssatzes *InterestRate* aufgebauten Darlehens *ResultLoan*. Der Kapitalwert kann von Null abweichen, wenn *MarginAdjustment* auf *False* gestellt war und wenn das Verfahren des tilgungsschichten-gewichteten Einstandes verwendet wurde.

11.2.9 ResultLoan

Verwendung:	Darlehen mit ermitteltem Zinssatz
Datentyp:	BALoan
Lese-/Schreibzugriff:	Ja/Nein

Nach erfolgreichem Aufruf von *Calculate* enthält *ResultLoan* das Darlehen, das aus dem ermittelten Zinssatz *InterestRate* (incl. Marge) aufgebaut wurde.

11.2.10 ResultWeightedAmortisation

Verwendung:	Berechnungsnachweis für den tilgungsschichten-gewichteten Einstand
Datentyp:	baDataArray
Lese-/Schreibzugriff:	Ja/Nein

Wurde die Berechnungsmethode „tilgungsschichten-gewichteter Einstand“ verwendet, enthält *ResultWeightedAmortisation* nach erfolgreichem Aufruf von *Calculate* einen entsprechenden Berechnungsnachweis mit den folgenden Daten:

- Datum (*baHeadDate*)
- Laufzeit in Jahren (*baHeadMaturity*)
- Tilgungszahlung (*baHeadAmortisationPayment*)
- Relative Tilgung (*baHeadRelativeAmortisation*)
  (Tilgungszahlung / Summe aller Tilgungszahlungen)
- Gewichtete Tilgung (*baHeadWeightedAmortisation*)
  (Relative Tilgung * Laufzeit)
- Zinssatz (*baHeadInterestRate*)
- Gewichteter Zins (*baHeadWeightedInterest*)
  (Gewichtete Tilgung * Zinssatz)

11.2.11 Tag

Verwendung:	Frei verwendbares Merkmal
Datentyp:	Variant
Lese-/Schreibzugriff:	Ja/Ja
Eingabe erforderlich:	Nein

11.3 Methoden

11.3.1 Calculate

Funktion:	Ermittlung des Zinssatzes
Parameter:	Keine
Rückgabewert:	Keiner

Berechnet den Zinseinstand bzw. den Zinssatz.

11.4 Ereignisse

11.4.1 AfterCalculation

Auslösung:	Nach erfolgreichem Abschluss der Methode Calculate
Parameter:	Keiner

11.4.2 PropertyChanged

Auslösung:	Nach Veränderung einer berechnungsrelevanten Eigenschaft
Parameter:	Keiner

12 Klasse BAAssistant

12.1 Aufgaben & Anwendung

Die Klasse *BAAssistant* bietet grundlegende Funktionen zur Unterstützung der anderen Klassen. In der aktuellen Version sind lediglich zwei Funktionen zur Interpolation von Daten sowie eine Funktion zur Rückgabe von Konstantenbezeichnungen implementiert.

12.2 Eigenschaften

BAAssistant verfügt über keine Eigenschaften.

12.3 Methoden

12.3.1 CubicSpline

Funktion:	Interpolation mittels einer Cubic Spline
Parameter:	X (Array of Variant)
	Y (Array of Variant)
	Result (Array of Variant)
Rückgabewert:	Boolean

CubicSpline interpoliert Werte zwischen den in *X()* und *Y()* definierten Originalwerten mit Hilfe von mehreren Polynomen dritten Grades und gibt die Ergebnisse über *Result()* zurück. War der Aufruf erfolgreich, gibt *CubicSpline* den Wert *True* zurück, andernfalls *False*. Eine Extrapolation ist nicht möglich.

- **X()**

  X() ist als eindimensionaler Array zu übergeben, der die X-Werte derjenigen Orginaldatenpunkte enthält, anhand derer die Interpolation erfolgen soll. *X()* muss genau so viele Elemente enthalten wie *Y()*.

- **Y()**

  Y() ist als eindimensionaler Array zu übergeben, der analog zu X() die Y-Werte der Originaldatenpunkte enthält. Y() muss genau so viele Elemente enthalten wie X().

- **Result()**

  Result() wird als zweidimensionaler Array übergeben, wobei das erste Feld die X-Werte enthält, zu denen der entsprechende Y-Wert interpoliert werden soll. Nach erfolgreichem Aufruf von *CubicSpline* ist das zweite Feld mit den ermittelten Y-Werten befüllt.

Die Spline wird anhand folgender Kriterien definiert:

- Die Spline besteht aus Polynomen dritten Grades (Cubic)

  $f(x) = a + bx + cx^2 + dx^3$
- Jedes Polynom geht exakt durch die es bestimmenden Punkte
- Zwei an einen Punkt grenzende Polynome weisen in diesem Punkt die gleiche erste Ableitung auf
- Die erste Ableitung in einem Punkt ermittelt sich aus der Steigung einer Geraden jeweils vom vorhergehenden zum nachfolgenden Punkt.
- Die zweite Ableitung im ersten und letzten Punkt ist 0

12.3.2 GetConstantDescription

Funktion:	Liefert eine sprechende Bezeichnung für eine Konstante zurück
Parameter:	babzConstant (Variant)
	ConstantType (Byte)
	Unknown (String, optional)
Rückgabewert:	String

Mit *GetConstantDescription* können in ein Anwenderprogramm sprechende Bezeichnungen der babz-spezifischen Konstanten eingebunden werden. Für *babzConstant* wird die Konstante übergeben, deren Bezeichnung ermittelt werden soll. Da die Konstanten hinsichtlich ihres numerischen Werts nicht eindeutig sind, muss mit *ConstantType* noch angegeben werden, aus

welcher Kategorie die Konstante stammt (siehe nachfolgende Tabelle). Für *Unknown* kann optional ein Wert angegeben werden, der zurückgemeldet wird, wenn *babzConstant* nicht bekannt ist (Standard: '[?]').

Konstante	**Wert**	**Hinweis**
baConstType_BankHolidays	1	Bankfeiertage
baConstType_Calendars	2	Vordefinierte Kalender
baConstType_DayCountConventions	3	Day Count Conventions
baConstType_BusinessDayConventions	4	Business Day Conventions
baConstType_TimeUnits	5	Zeiteinheiten
baConstType_InterpolationModes	6	Interpolationsmethoden
baConstType_Interestrates	7	Zinssatztypen (Geld, Swap, interpoliert)
baConstType_AutoFixedLeg	8	Arten der automatischen Erstellung des fixed Leg bei Zinsswaps
baConstType_Amortisation	9	Tilgungsarten
baConstType_InterestMethods	10	Verzinsungsmethoden
baConstType_ManualAmortisation	11	Manuelle Amortisation: Addieren, Ersetzen
baConstType_ManualInterest	12	Manuelle Zinsen: Addieren, Ersetzen, ...
baConstType_Events	13	Ereignisse
baConstType_Headings	14	Überschriften/Datentypen (BAOutput)
baConstType_InterestPayment	15	Zinsbeibringung bei Annuitätendarlehen
baConstType_LoanPricingModes	16	Methoden der Einstandsermittlung
baConstType_InterestOrigins	17	Herkunft des Zinssatzes
baConstType_InterestSwapTypes	18	Zinsswap-Arten
baConstType_RateTypes	19	Satz-/Kurstyp (Bid, Mid, Ask)
baConstType_InterestSwapValuationModes	20	Art des Zinssswap-Bewertung (Brutto, Netto)
baConstType_FXForwardValuationModes	21	Art der Bewertung von Devisentermingeschäften (Neuabschluss, Gegengeschäft, Mittelkurs)

12.3.3 LinearInterpolation

Funktion:	Lineare Interpolation
Parameter:	X (Array of Variant)
	Y (Array of Variant)
	Result (Array of Variant)
Rückgabewert:	Boolean

LinearInterpolation interpoliert Werte zwischen den in *X()* und *Y()* definierten Originalwerten mit Hilfe mehrerer Geraden und gibt die Ergebnisse über *Result()* zurück. War der Aufruf erfolgreich, gibt *LinearInterpolation* den Wert *True* zurück, andernfalls *False*. Eine Extrapolation ist nicht möglich.

1. **X()**

   X() ist als eindimensionaler Array zu übergeben, der die X-Werte derjenigen Orginaldatenpunkte enthält, anhand derer die Interpolation erfolgen soll. *X()* muss genau so viele Elemente enthalten wie *Y()*.

2. **Y()**

   Y() ist als eindimensionaler Array zu übergeben, der analog zu X() die Y-Werte der Originaldatenpunkte enthält. Y() muss genau so viele Elemente enthalten wie X().

3. **Result()**

   Result() wird als zweidimensionaler Array übergeben, wobei das erste Feld die X-Werte enthält, zu denen der entsprechende Y-Wert interpoliert werden soll. Nach erfolgreichem Aufruf von *LinearInterpolation* ist das zweite Feld mit den ermittelten Y-Werten befüllt.

Bei der linearen Interpolation handelt es sich um eine lineare Spline. Die einzelnen Punkte sind durch Gerade verbunden.

13 Interface für Microsoft Excel®

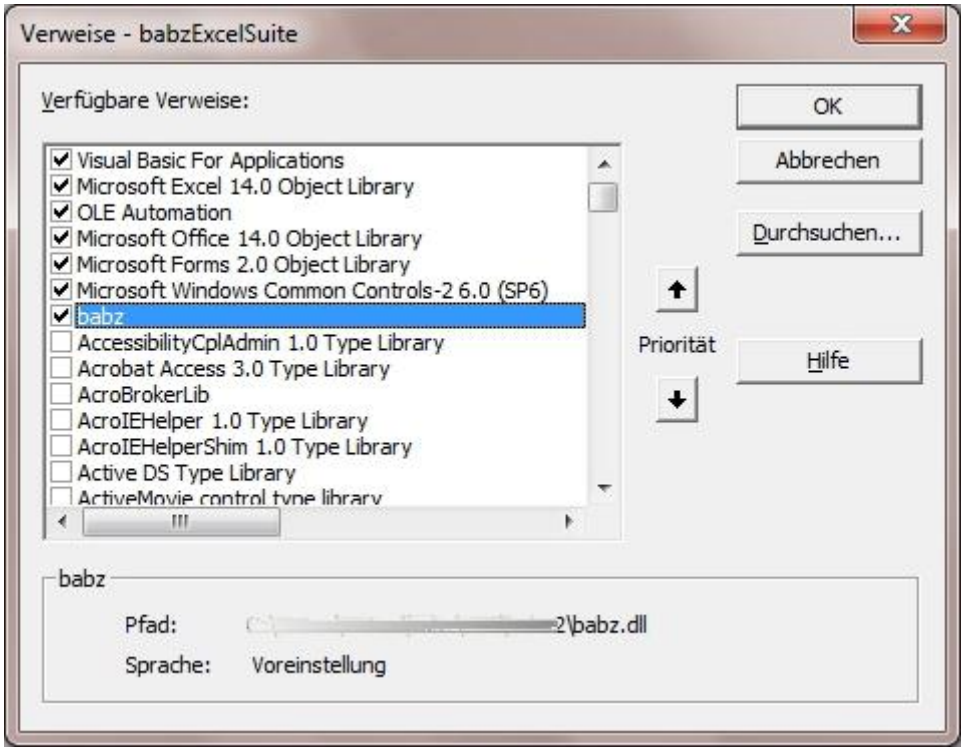
13.1 Allgemeines

Babz verfügt mit Ausnahme der Dialoge für den Registrierungsschlüssel und die Systemeinstellungen über keinerlei Schnittstellen, die dem Anwender unmittelbar zur Eingabe oder Darstellung von Daten dienen. Dazu sind Anwenderprogramme erforderlich, die Aufgaben wie die Eingabe, Verwaltung und Speicherung von Daten, die Darstellung und grafische Aufbereitung von Berechnungsergebnissen, die Bereitstellung von Marktdaten etc. übernehmen. Babz wird – zumindest derzeit – ohne ein solches Anwenderprogramm ausgeliefert. Um die Funktionen von babz dennoch schnell und einfach zugänglich zu machen, wird eine Anbindung („Interface“) an Microsoft Excel® bereitgestellt, die einfache Funktionalitäten bereitstellt und darüber hinaus ein Beispiel zum Zugriff auf den babz-Rechenkern darstellt.

Das Interface für Excel® besteht aus einer Arbeitsmappe mit VBA-Code, die ab Excel® 97 funktionsfähig ist. Mit Hilfe eines Menüs können der Arbeitsmappe verschiedene „Objekte“ (z.B. Zinskurven, Darlehen) hinzugefügt werden, die nach der Befüllung mit Daten und Parametern berechnet werden können. Der Programmcode ist frei zugänglich, so dass das Interface an die eigenen Anforderungen und Wünsche angepasst werden kann. Zu beachten ist, dass über das Interface nicht alle Funktionalitäten des babz-Rechenkerns zugänglich sind, so können z.B. nur vordefinierte Kalender verwendet werden.

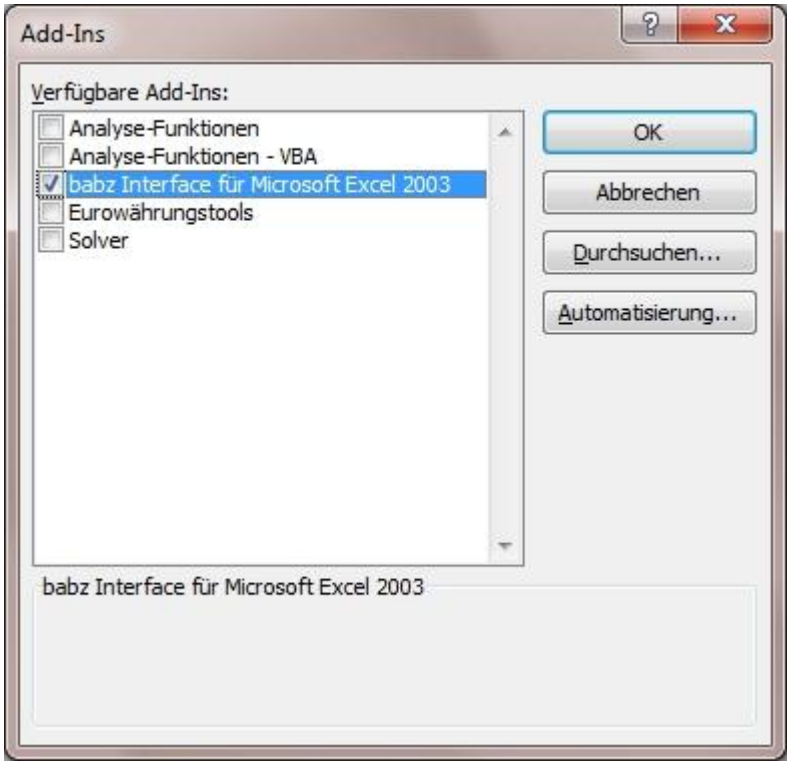
13.2 Einrichtung

Für das Interface ist keine Installation erforderlich, es muss lediglich der babz-Rechenkern auf dem Rechner registriert sein (siehe „Anhang II Registrierung der Datei babz.dll“ auf Seite 153). Ob der Verweis besteht, können Sie im Visual Basic Editor prüfen (Aufruf des Visual Basic Editors mit <Alt> + <F11>, Menü „Extras“, Punkt „Verweise …“).



Damit das Interface funktioniert, müssen Makros zugelassen sein. Aus Sicherheitsgründen ist es nicht ratsam, Makros grundsätzlich zuzulassen. Eine mögliche Lösung ist, ein Verzeichnis auf Ihrem Computer als sicheren Speicherort zu kennzeichnen und dort nach erfolgter Prüfung auf Viren und Schadcode das Interface abzulegen.

In der ausgelieferten Version als eigenständige Microsoft Excel®-Datei muss das Interface vor der Durchführung von Berechnung geöffnet werden, nach dem Schließen stehen die Funktionen von babz nicht mehr zur Verfügung. Darüber hinaus besteht der Nachteil, dass die Speicherung der erstellten Tabellenblätter/Objekte und des VBA-Codes in einer Datei erfolgt, was den Austausch des Interface gegen eine neuere Version erschwert. Es empfiehlt sich daher, die Interface-Datei als Add-In zu speichern (Menü „Datei/Speichern unter …“) und entsprechend in Microsoft Excel® einzubinden. Babz steht dann unabhängig von der geöffneten Arbeitsmappe zur Verfügung. Wie Sie Add-Ins einbinden, entnehmen Sie bitte der Dokumentation zu Microsoft Excel® (bei älteren Versionen Menü „Extras/Add-Ins …“, bei neueren Versionen „Datei/Optionen/Add-Ins/Gehe zu …“).



13.3 Aufbau

Die Programmierung des Interface soll an dieser Stelle nicht ausführlich erläutert werden, da sie sich am einfachsten durch die Ansicht des VBA-Codes nachvollzogen werden kann. Das zentrale Modul „xlBABZInterface“ enthält im Wesentlichen für jede babz-Objektklasse je eine Prozedur zum Einlesen der auf einem Tabellenblatt gespeicherten Daten (Read_xl…), zum Berechnen (Recalc_xl…) und zum Ausgeben der Berechnungsergebnisse (Write_xl…). Die weiteren Funktionen und Prozeduren dienen der Erstellung von Formatvorlagen u.ä.

13.4 Bedienung

Beim Öffnen der Interface-Datei wird eine neue Menüleiste erstellt, die den Zugriff auf die verschiedenen Funktionen ermöglicht. Durch Klick auf einen der Punkte im Menü „Neu“ können der Arbeitsmappe neue Tabellenblätter zur Darstellung der verschiedenen babz-Objektklassen hinzugefügt werden. Mit „Duplizieren“ können bestehende Tabellenblätter kopiert und als neues Tabellenblatt eingefügt werden. „Berechnen“ liest die auf einem Tabellenblatt gespeicherten

Parameter ein, übergibt sie zur Berechnung an den babz-Rechenkern und schreibt die Ergebnisse wieder in das Tabellenblatt.

Das Menü „Rechenkern“ bietet die Möglichkeit, einen Registrierungsschlüssel einzugeben, Einstellungen des Rechenkerns zu verändern und evtl. mitgeschnittene Ereignisse in eine Textdatei zu schreiben.

Die Tabellenblätter für die verschiedenen Klassen sind jeweils ähnlich aufgebaut. Im oberen Bereich finden sich die Einstellungen und Berechnungsparameter, im unteren Bereich werden die Ergebnisse ausgegeben. Farblich hervorgehobene „Value“-Zellen dienen der Eingabe von Berechnungsparametern, die anderen Zellen werden nicht verarbeitet. Ein Doppelklick auf Zellen, die der Eingabe eines Darlehens oder einer Zinskurve dienen, öffnet einen Auswahldialog. Bei einigen Zellen ist die Eingabemöglichkeit auf für die jeweilige Eigenschaft gültige Werte eingeschränkt. In der Spalte „Hint“ wird die sprechende Bezeichnung des gewählten Parameters angezeigt.

Der Aufbau der Tabellenblätter darf nicht verändert werden, damit das Interface erkennen kann, welche Klasse auf dem jeweiligen Tabellenblatt gespeichert ist. Bitte beachten Sie auch, dass alle Inhalte, die nicht in einer farblich markierten Zelle stehen, bei Klick auf „Berechnen“ verloren gehen.

Loan: New Loan

Loan Settings

Property	Value	Hint
Description	New Loan	
Amount	1.200.000,00	
Disagio	0,000%	
Remainder	0,00	
Calendar 1	1	Target
Calendar 2		None
Drawing\Amount	1.200.000,00	
Drawing\Date	Sa, 01.01.2011	
Drawing\BDC	2	Following, Unadjusted
Amortisation\First	Mo, 31.01.2011	
Amortisation\Ultimo	WAHR	
Amortisation\Interval	1	
Amortisation\BDC	132	Modified Following, Adjusted
Amortisation\Count	12	
Amortisation\Last	Fr, 30.12.2011	

Die im Interface verwendeten Farben ändern sich abhängig von der verwendeten Excel®-Version, da niedrigere Versionen weniger Farben darstellen können. Bis einschließlich Excel® 2003 wird ein eingeschränktes Farbspektrum verwendet, darüber ein größeres Farbspektrum.

Anhang I Laufzeitfehler

Im Falle ungültiger Zustände, Berechnungsergebnisse und Berechnungsparameter zur Laufzeit werde die in nachfolgender Tabelle aufgelisteten Fehler ausgelöst, die Rückschlüsse auf die Fehlerquelle zulassen.

Die Fehlernummer eines Laufzeitfehlers setzt sich zusammen aus der entsprechenden Nummer, wie sie in der Tabelle aufgeführt ist, dem Wert '16.000' als Basis für Laufzeitfehlers des babz-Rechenkerns und dem Wert '-2.147.221.504' (vbObjectError).

Beispiel: *Tritt ein Fehler mit der Fehlernummer '-2.147.205.503' auf, so handelt es sich um den Laufzeitfehler mit der babz-internen Fehlernummer '1' (Der Ostersonntag konnte nicht bestimmt werden).*
-2.147.221.504 + 16.000 + 1 = -2.147.205.503

Sofern eine Identifizierung der Fehlerquelle anhand des Laufzeitfehlers nicht möglich ist, aktivieren Sie den *Trace*, um den Fehler näher zu untersuchen. Nähere Ausführungen zur *Trace*-Funktionalität finden Sie im Kapitel „2.2.10 Trace„ auf Seite 23.

Nummer	Beschreibung	Auslösende Klasse Hinweis
0	Unbekannter Fehler	Alle Klassen
1	Der Ostersonntag konnte nicht bestimmt werden	BACalendar
2	Unbekannter Kalender	BACalendar Es wurde versucht, mit der Methode 'Load' eines Objektes der Klasse 'BACalendar' einen unbekannten Kalender zu laden.
3	Unbekannter vordefinierter Bankfeiertag	BACalendar Die Auflistung 'Holidays' eines Objektes der Klasse 'BACalendar' enthält einen Eintrag vom Typ 'baHoliday', dessen Element 'Predefined' auf einen ungültigen Feiertag verweist (bzw. es wurde versucht, einen solchen Feiertag mit der

		Methode AddPredefinedHoliday festzulegen).
4	Die Collection 'Holidays' darf nur Werte des Typs 'baHoliday' enthalten	BACalendar Die Auflistung 'Holidays' eines Objektes der Klasse 'BACalendar' enthält mindestens einen Eintrag, der nicht dem Typ 'baHoliday' entspricht.
5	Ein an den Ostersonntag geknüpfter Feiertag liegt ausserhalb des geprüften Kalenderjahres	BACalendar Die Auflistung 'Holidays' eines Objektes der Klasse 'BACalendar' enthält einen beweglichen Feiertag, der so weit vor oder nach dem Ostersonntag eines Kalenderjahres liegt, dass das Kalenderjahr verlassen wird.
6	Ein Element der Holidays-Auflistung enthält ein ungültiges Datum	BACalendar Die Auflistung 'Holidays' eines Objektes der Klasse 'BACalendar' enthält einen festen Feiertag, dessen Element 'Day' nicht zwischen 1 und 31 liegt (jeweils einschließlich), und/oder dessen Element 'Month' nicht zwischen 1 und 12 liegt (jeweils einschließlich).
7	Unbekannte Zeiteinheit	BACalendar Es wurde versucht, mit der Methode 'AddDateInterval' eines Objektes der Klasse 'BACalendar' eine unbekannte Zeiteinheit zu verwenden.
8	Unbekannte Day Count Convention	BACalendar
9	Keine Basisdaten für Interpolation vorhanden	BAInterestCurve Es wurden keine geeigneten Zinssätze gefunden, anhand derer für das Bootstrapping benötigte Zinssätze interpoliert werden könnten. Möglicherweise sind die vorhandenen Zinssätze ihrerseits bereits interpoliert und werden daher nicht herangezogen.
10	Ein benötigter Wert konnte nicht interpoliert werden	BAInterestCurve
11	Unbekannte Interpolationsart	BAInterestCurve
12	Kein Swapsatz vorhanden	BAInterestCurve
13	Ein Datum liegt außerhalb der Zinskurve	BAInterestCurve Es wurde ein Zinssatz oder Diskontfaktor eines Objektes der Klasse 'BAInterestCurve' abgefragt, der vor dem Settlement-Datum der Zinskurve oder nach der längsten verfügbaren Laufzeit liegt.
14	Daten liegen in der falschen Reihenfolge	BAInterestCurve

	vor (Stopp vor Start)	Für ein Objekt der Klasse 'BAInterestCurve' wurde mit der Funktion 'ForwardRate' ein Forward-Satz abgefragt, wobei 'EndDate' zeitlich vor 'StartDate' liegt.
15	Fehler bei der Ermittlung des Diskontfaktors	BAInterestCurve
16	Ungültiger Disagio-Wert (muss zwischen 0 und 1 liegen)	BALoan
17	Ungültiger Ziehungsbetrag	BALoan Wird für ein Objekt der Klasse 'BALoan' kein Kreditbetrag angegeben ('Amount'), muss der Ziehungsbetrag einen Wert > Null aufweisen ('Drawing.Amount').
18	Ungültiger Restsaldo (muss zwischen 0 und Kreditbetrag liegen)	BALoan
19	Anzahl Tilgungstermine < 0	BALoan
20	Anzahl Zinstermine < 0	BALoan
21	Unbekannte Verzinsungsmethode	BALoan Für ein Objekt der Klasse 'BALoan' wurde eine unbekannte Verzinsungsmethode gewählt ('Interest.InterestMethod').
22	Unbekannte Darlehensart	BALoan Für ein Objekt der Klasse 'BALoan' wurde eine unbekannte Darlehensart gewählt ('Amortisation.Type').
23	Fehler beim Ausrollen des Zeitgerüsts	BALoan
24	Unbekannte manuelle Tilgung	BALoan Die Auflistung 'ManualAmortisation' eines Objektes der Klasse 'BALoan' enthält einen Eintrag vom Typ 'baManualAmortisation', dessen Element 'Type' auf einen ungültigen Wert gestellt ist.
25	Die Collection 'ManualAmortisation' darf nur Werte des Typs 'baManualAmortisation' enthalten	BALoan Die Auflistung 'ManualAmortisation' eines Objektes der Klasse 'BALOan' enthält mindestens einen Eintrag, der nicht dem Typ 'baManualAmortisation' entspricht
26	Die Spalte mit den Terminen wurde im DataArray nicht gefunden	BAOutput Ein Objekt der Klasse 'BAOutput' enthält einen DataArray, der über keine Spalte des Datentyps 'baHeadDate' verfügt. Für eine Berechnung wird diese Datenspalte jedoch benötigt.
27	Eine erforderliche Spalte wurde nicht im	BAOutput

	DataArray gefunden	Ein Objekt der Klasse 'BAOutput' enthält einen DataArray, in dem eine Spalte nicht vorhanden ist, die für eine Berechnung benötigt wird.
28	Im DataArray wurde keine Zeile gefunden, die nach dem Settlement der Zinskurve liegt	BAOutput BAPrepaymentCompensation In einem DataArray eines Objektes der Klasse 'BAOutput' sind nach einem gegebenen Datum keine Werte vorhanden.
29	Keine Zinskurve vorhanden oder Fehler bei der Berechnung der Zinskurve	Mehrere Klassen Für eine Berechnung wird ein Objekt der Klasse 'BAInterestCurve' benötigt. Dieses Objekt fehlt oder der Aufruf der Methode 'Calculate' ist fehlgeschlagen.
30	Der Restsaldo zum Ablösedatum konnte nicht ermittelt werden	BAPrepaymentCompensation Bei der Ermittlung der zum Ablösestichtag fälligen Zahlungen (Restsaldo + Stückzinsen) ist ein Fehler aufgetreten.
31	Variabel verzinsliche Darlehen werden nicht unterstützt	BAPrePaymentCompensation
32	Das Settlement-Datum der Zinskurve darf nicht nach dem Ziehungsdatum des Darlehens liegen	BALoanPricing
33	Es konnte nicht ermittelt werden, wann eine Periode, für die ein Forward-Satz ermittelt werden sollte, endet	BALoan
34	[Frei]	
35	Die Collection 'DateSum' darf nur Werte des Typs 'baDateSum' enthalten	BAOutput Die Auflistung 'DateSum' eines Objektes der Klasse 'BAOutput' enthält mindestens einen Eintrag, der nicht dem Typ 'baDateSum' entspricht.
36	Fehler bei der Vorbereitung des variabel verzinslichen Darlehens (Floating Leg)	BAInterestSwap Beim Aufruf der Methode 'Calculate' eines Objektes der Klasse 'BALoan', das die variabel verzinsliche Seite eines Zinsswaps beschreibt, ist ein Fehler aufgetreten.
37	Fehler bei der Vorbereitung des fest verzinslichen Darlehens (Fixed Leg)	BAInterestSwap Beim Aufruf der Methode 'Calculate' eines Objektes der Klasse 'BALoan', das die festverzinsliche Seite eines Zinsswaps beschreibt, ist ein Fehler aufgetreten.
38	Das als Floating Leg übergebene Darle-	BAInterestSwap

	hen ist festzverzinst	Bei einem Objekt der Klasse 'BALoan', das die variabel verzinsliche Seite eines Zinsswaps beschreibt, ist die variable Verzinsung nicht aktiv (das Darlehen ist festverzinst).
39	Die Settlement- und Fixing-Daten der Payer-Zinskurve, der Receiver-Zinskurve und der Floating-Zinskurve des Darlehens müssen identisch sein	BAInterestCurve
40	Eine erforderliche Zeile wurde nicht im DataArray gefunden	BAOutput Ein Objekt der Klasse 'BAOutput' enthält einen DataArray, in dem eine Zeile nicht vorhanden ist, die für eine Berechnung benötigt wird.
41	Der erste Zinsanpassungstermin konnte nicht bestimmt werden (Auto First Float)	BALoan
42	Das Fixing-Datum der Zinskurve darf nicht nach dem Systemdatum liegen	BAInterestCurve
43	Unbekannter Berechnungsmodus	BALoanPricing Die Eigenschaft 'CalculationMode' eines Objektes der Klasse BALoanPricing enthält einen ungültigen Wert.
44	Keine Trace-Daten vorhanden	BASystem Die Methode 'WriteTraceToFile' konnte nicht durchgeführt werden, da die Auflistung 'Trace' keine Elemente enthält.
45	Die Collection 'ManualInterest' darf nur Werte des Typs 'baManualInterest' enthalten	BALoan Die Auflistung 'ManualInterest' eines Objektes der Klasse 'BALoan' enthält mindestens einen Eintrag, der nicht dem Typ 'baManualInterest' entspricht
46	'baInterestAdditional' und 'baInterestReplacement' dürfen nicht gleichzeitig verwendet werden	BALoan Die Auflistung 'ManualInterest' eines Objektes der Klasse 'BALoan' enthält einen Eintrag, dessen Element 'Type' gleichzeitig die Konstanten 'baInterestAdditional' und 'baInterestReplacement' enthält.
47	'baInterestAdditional' und 'baInterestAddMargin' dürfen nicht gleichzeitig verwendet werden	BALoan Die Auflistung 'ManualInterest' eines Objektes der Klasse 'BALoan' enthält einen Eintrag, dessen Element 'Type' gleichzeitig die Konstanten 'baInterestAdditional' und 'baInterestAddMargin' enthält. 'baInterestAddMargin' ist nur in

		Verbindung mit 'baInterestReplacement' möglich.
48	Beim Zugriff auf die Windows-Registry ist ein Fehler aufgetreten	BASystem Beim Lesen und Schreiben des Registrierungsschlüssels in oder aus der Windows-Registry ist ein Fehler aufgetreten.
49	Ein vorgegebener n. Wochentag konnte nicht im angegebenen Kalendermonat ermittelt werden	BACalendar Bei der Bestimmung eines variablen Feiertags, der an einen bestimmten Wochentag gekoppelt ist, wurde der Kalendermonat verlassen (z.B. weil es keinen 8. Montag im Januar gibt).
50	Das Settlement-Datum fällt auf einen Bankfeiertag	BAInterestCurve
51	Unbekannte Konstanten-Kategorie	BAAssistant Der Funktion 'GetConstantDescription' wurde für 'ConstantType' ein unbekannter Wert übergeben.
52	Die Währung des Darlehens muss mit der Währung der Zinskurve übereinstimmen	Mehrere Klassen Die Währungen eines Darlehens und der diesem zugewiesenen Zinskurve stimmen nicht überein.
53	Keine Währung angegeben	Mehrere Klassen Für ein Objekt wurde keine Währung angegeben (Eigenschaft 'Curr'), die Systemeinstellung 'CurrRequired' ist jedoch auf 'True' gesetzt.
54	Die Währung der Darlehen muss übereinstimmen	Mehrere Klassen Für eine Berechnung sollen zwei Darlehen herangezogen werden, deren Währungen nicht übereinstimmen.
55	Die Währung der Zinskurven muss übereinstimmen	Mehrere Klassen Für eine Berechnung sollen zwei Zinskurven herangezogen werden, deren Währungen nicht übereinstimmen.
56	Der Zugriff auf die Windows-Registry ist untersagt (Systemeinstellung 'setting_NoRegKeyToRegistry')	
57	Abbruch durch Anwender	BASystem Die den Methoden ShowRegKeyDialog und ShowSettingsDialog aufgerufenen Anwenderdialoge lösen diesen Fehler aus, wenn der Anwender die Schaltfläche 'Abbrechen' klickt.
58	Die Receiver-Zinskurve konnte nicht berechnet werden	BAInterestSwap

59	Die Payer-Zinskurve konnte nicht berechnet werden	BAInterestSwao
60	Die Floating-Zinskurve konnte nicht berechnet werden	BALoan
61	Fehler bei der Ermittlung der ersten Zinsanpassung (AutoFloat)	BALoan
62	Fehler bei der Berechnung der Tilgungsrate/Annuität	BALoan
63	Fehler bei der variablen Verzinsung	BALoan
64	Fehler bei der manuellen Zinssteuerung	BALoan Ein in der Auflistung 'ManualInterest' hinterlegter Eintrag konnte nicht richtig umgesetzt werden.
65	Fehler bei der Zusammenstellung der Berechnungsergebnisse	Mehrere Klassen
66	Die Datumsdefinition entspricht nicht dem erwarteten Aufbau	BAOutput Die Funktion 'Merge' konnte das in einem Feld des Datentyps 'baHeadDate' oder 'baHeadValueDate' gespeicherte Datum nicht verwenden. Weisen Sie solchen Feldern nur Variablen des Typs 'Date' zu.
67	Die Währung der Zinskurve muss der lokalen Währung (CurrDomestic) entsprechen	BAFXForward
68	Das Währungspaar des Zinstermingeschäfts muss mit dem Währungspaar der Terminkurve übereinstimmen	BAFXForward Es können nur solche Devisentermingeschäfte bewertet werden, deren Währungspaar (CurrDomestic und CurrForeign) mit dem Währungspaar der Devisenterminkurve übereinstimmt.
69	Die Zahlungen in der lokalen Währung und in der Fremdwährung müssen unterschiedliche Vorzeichen aufweisen	BAFXForward Die Beträge in lokaler Währung und Fremdwährung eines Devisentermingeschäfts müssen unterschiedliche Vorzeichen aufweisen.
70	Es muss entweder ein Betrag in lokaler Währung oder ein Betrag in Fremdwährung angegeben werden	BAFXForward Ist für ein Devisentermingeschäft der fixierte Wechselkurs angegeben (FXRateFixed), muss entweder der Betrag in lokaler Währung (AmountDomestic) oder der Betrag in Fremd-

		währung (AmountForeign) angegeben werden.
71	Keine Terminkurve vorhanden oder Fehler bei der Berechnung der Terminkurve	BAFXForward
72	Der Termin-Wechselkurs konnte nicht ermittelt werden	BAFXForward Ein erforderlicher Devisenterminkurs konnte nicht aus der Devisenterminkurve ermittelt werden. Möglicherweise liegt der Termin außerhalb der Devisenterminkurve.
73	Der Fälligkeitstermin ist ein Feiertag im Kalender der Zinskurve	BAFXForward Der Fälligkeitstermin eines Devisentermingeschäfts fällt auf einen Feiertag im Kalender der Zinskurve. Der Fälligkeitstermin darf weder in diesem Kalender noch im Kalender der Devisenterminkurve auf einen Feiertag fallen.
74	Der Fälligkeitstermin ist ein Feiertag im Kalender der Terminkurve	BAFXForward Der Fälligkeitstermin eines Devisentermingeschäfts fällt auf einen Feiertag im Kalender der Devisenterminkurve. Der Fälligkeitstermin darf weder in diesem Kalender noch im Kalender der Zinskurve auf einen Feiertag fallen.
75	Kein Kalender vorhanden oder Fehler bei Zugriff auf den Kalender	Mehrere Klassen Der Eigenschaft 'Calendar' muss auch dann ein Objekt der Klasse 'BACalendar' zugewiesen werden, wenn keine Feiertage definiert werden sollen.
76	Die Auflistung 'Swaps' enthält mindestens zwei Elemente, die auf das gleiche Datum verweisen	BAFXForwardCurve
77	'Settlement' wurde manuell festgelegt. 'Fixing' und 'SettlementDays' können nicht verändert werden	BAInterestCurve Für Zinskurven, die mittels der Funktion 'GetForwardCurve' aus einer anderen Zinskurve abgeleitet wurden, sind die Eigenschaften 'Fixing' und 'SettlementDays' schreibgeschützt.
78	Constant-Maturity-Strukturen sind nicht zulässig (Systemeinstellung 'ConstantMaturityAllowed')	*Nicht dokumentiert – nicht zur Anwendung freigegeben.*
79	Die Eigenschaft 'ConstantMaturity' darf nur positive Werte enthalten	*Nicht dokumentiert – nicht zur Anwendung freigegeben.*
80	Ein als Fixed Leg übergebenes Darlehen ist variabel verzinst	BAInterestSwap Das Darlehen, welches das Fixed Leg (LoanFixed) definiert, ist

		variabel verzinst.
81	'AutoFixedLeg' muss Null sein bei Zinsswaps mit zwei Floating Legs	BAInterestSwap
82	Spot-Wechselkurse: Es muss entweder Bid- und Ask-Kurs, Bid- und Mid-Kurs oder Ask- und Mid-Kurs angegeben werden	BAFXForwardCurve Es müssen zwei Kursarten angegeben werden, damit alle drei Kursarten Mid, Bid und Ask berechnet werden können.
83	Ungültige Kursart (zulässig: Bid, Ask, Mid)	BAFXForwardCurve Die Funktion 'GetOutright' kann nur die aufgeführten Kursarten zurückgeben.
84	Für einen Datentyp im DataArray ist nicht bekannt, wie dieser kombiniert werden kann	BAOutput Die Funktion 'Merge' kann nur solche Objekte der Klasse 'BAOutput' zusammenfassen, die babz-interne Datentypen enthalten, da für extern definierte Datentypen nicht bekannt ist, ob und wie diese zusammengeführt werden können.
85	Die Objekte können nicht zusammengefasst werden, da nicht beide entweder den Datentyp 'baHeadDate' oder 'baHeadValueDate' enthalten	BAOutput Sollen zwei Objektes der Klasse 'BAOutput' mit der Funktion 'Merge' zusammengefasst werden, müssen entweder beide Objekte den Datentyp 'baHeadDate' oder 'baHeadValueDate' enthalten, da diese als Sortierkriterium fungieren.
86	Die Settlement-Termine der Zinskurve und der Devisenterminkurve müssen übereinstimmen	BAFXForward
87	Die Eigenschaft 'LoanFixed' enthält kein Darlehen, obwohl 'AutoFixedLeg' nicht aktiv ist	BAInterestSwap Wenn das Fixed Leg nicht automatisch aus dem Floating Leg abgeleitet werden soll ('AutoFixedLeg = 0'), muss das Darlehen des Fixed Leg explizit angegeben werden.
88	Für eine Periode wurde kein Zinssatz angegeben oder konnte kein Zinssatz ermittelt werden	BALoan Die 'ManualInterest'-Auflistung enthält einen Zinssatz für eine Zinsperiode, für die nachfolgende Periode ist jedoch weder ein Zinssatz in der 'ManualInterest'-Auflistung enthalten noch wird für diese Periode ein Zinsterminsatz berechnet (Interest.Float). Passen Sie die variable Verzinsung an oder fügen Sie der 'ManualInterest'-Auflistung einen entsprechenden Eintrag hinzu oder fügen Sie dem vorhergehenden 'ManualInterest'-Eintrag die Kennzeichnung 'baInterestCarryOverAllowed' hinzu.

89	Wenn für eine Terminfolge das Merkmal 'Ultimo' gesetzt ist, muss der erste Termin der Folge auf einen Monatsletzten fallen	BALoan Wenn 'Ultimo' auf 'Wahr' gesetzt wurde, muss 'First' auf den letzten Tag des jeweiligen Kalendermonats fallen. Passen Sie 'First' an.
90	Die Terminanzahl 'Count' konnte nicht korrekt aus dem ersten und letzten Termin der Folge ermittelt werden.	BALoan Entweder ist der berechnete Wert für 'Count' ungültig (<0) oder die Verprobung ('Last' muss mit 'First + Interval x Count' übereinstimmen. Möglicherweise liegt 'Last' außerhalb der Terminreihe, die sich aus 'First' und 'Interval' ergibt.
91	Beim Aufruf eines User Exits ist ein Fehler aufgetreten	BAInterestCurve Prüfen Sie, ob der Verweis auf das UserExit-Objekt korrekt ist und das Objekt richtig funktioniert.
92	Eine Zinskurve darf nicht Cash-/Swap-Sätze und Zero-Sätze gleichzeitig enthalten	BAInterestCurve Prüfen Sie die in der Interestrates-Auflistung enthaltenen Sätze und stellen Sie sicher, dass entweder nur Zero-Sätze oder nur Geldmarkt- und Swapsätze enthalten sind.
93	Ungültiger Bewertungstyp	BAInterestSwap Für ValuationMode wurde ein ungültiger Wert gewählt.
94	Fehler bei der Saldierung von Zahlungsströmen	BAInterestSwap Bei der Bewertung nach der Netto-Methode konnten die Zahlungsströme der beiden Legs nicht korrekt saldiert werden, z.B. weil die Zahlungstermine nicht übereinstimmen.
95	'baInterestCalculateRate' darf nicht gleichzeitig mit 'baInterestAddMargin' oder 'baInterestAdditional' verwendet werden	BALoan Die Auflistung 'ManualInterest' eines Objektes der Klasse 'BALoan' enthält einen Eintrag, dessen Element 'Type' gleichzeitig die Konstanten 'baInterestCalculateRate' und 'baInterestAddMargin' oder 'baInterestAdditional' enthält.
100	Kein gültiger Registrierungsschlüssel	BASystem Der angegebene Registrierungsschlüssel ist ungültig. Einige Funktionen des Rechenkerns können nicht verwendet werden.

Anhang II Registrierung der Datei babz.dll

Die Funktionen des babz-Rechenkerns sind in der Datei „babz.dll“ untergebracht. Damit Anwendungen auf die Funktionen zugreifen können, muss die Datei „babz.dll“ im Windows®-System registriert werden. Der Begriff „Registrierung“ bezeichnet dabei lediglich die technische Einbindung des babz-Rechenkerns in das Windows®-System, nicht zu verwechseln mit der Eingabe eines Registrierungsschlüssels (Lizenzschlüssels) zur Freischaltung des Rechenkerns.

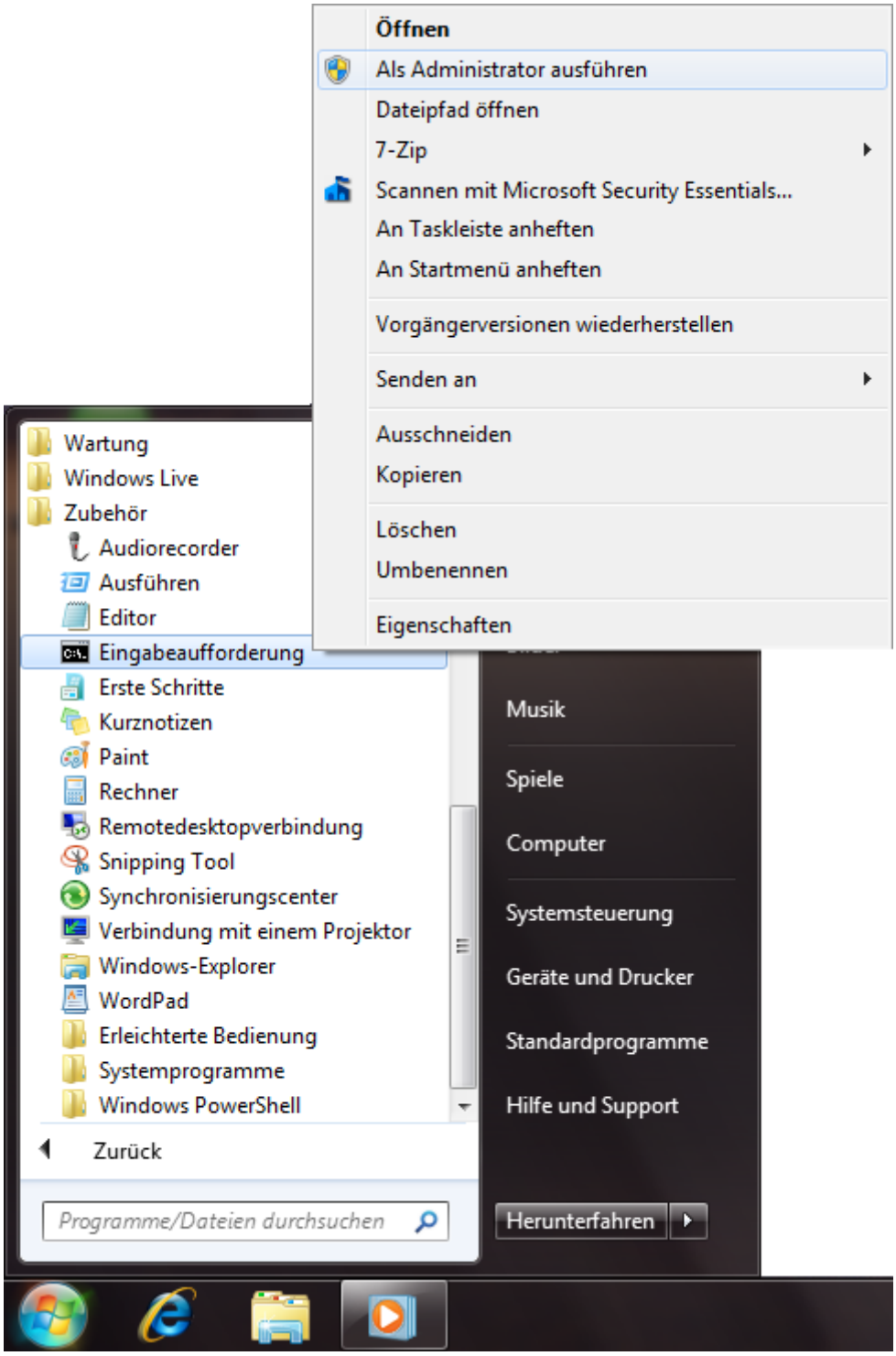
Die Registrierung muss, da babz nicht mit einem Installationsprogramm ausgeliefert wird, vom Anwender vorgenommen werden. Die einzelnen Schritte sind nachfolgenden für Windows 7® beschrieben. Die Registrierung unter Windows Vista® und Windows XP® läuft entsprechend ab.

1. Abspeichern der Datei „babz.dll“

Kopieren Sie die Datei „babz.dll“ und weitere Dateien des babz-Pakets (Handbücher) nach dem Download in einen Ordner auf der Festplatte Ihres Computers (z.B. 'C:\babz')

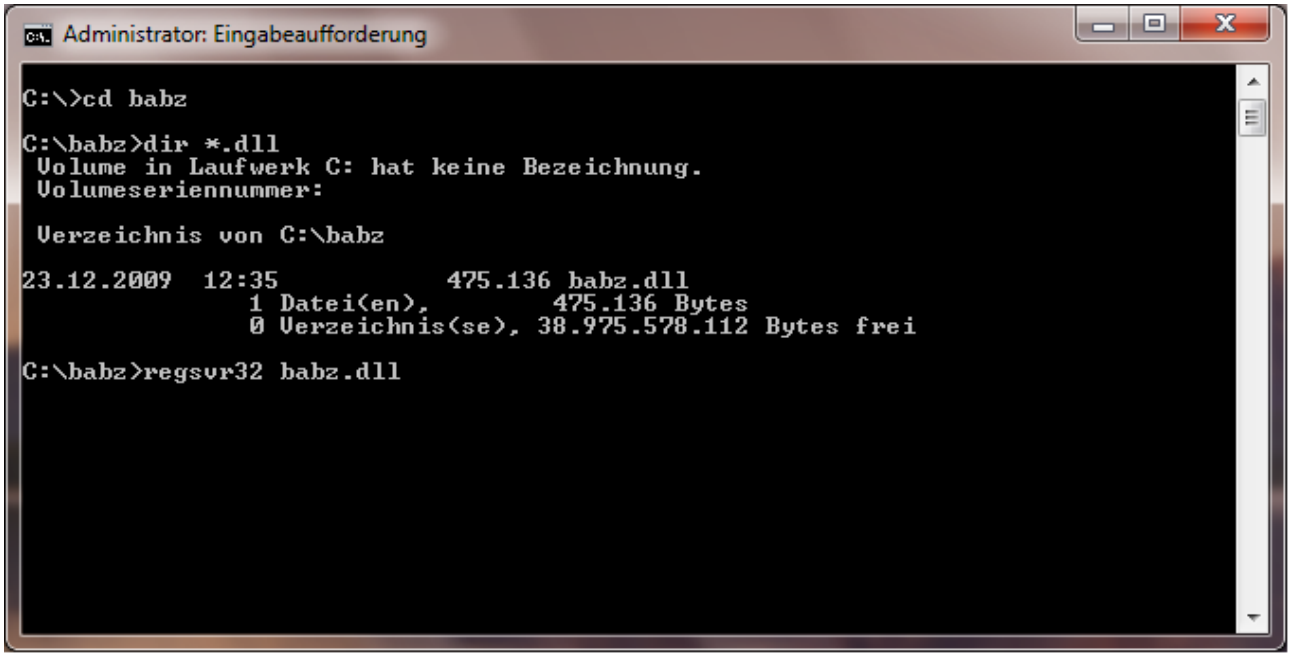
2. Aufruf der Eingabeaufforderung

Rufen Sie im Startmenü unter „Programme“ im Ordner „Zubehör“ die Eingabeaufforderung auf. Klicken Sie dazu unter Windows 7® und Windows Vista® mit der rechten Maustaste auf den Eintrag, wählen Sie „Als Administrator ausführen“ und bestätigen Sie die nachfolgende Sicherheitsabfrage mit „Ja“. Unter Windows XP® genügt es, wenn Sie mit Administratorenrechten angemeldet sind.

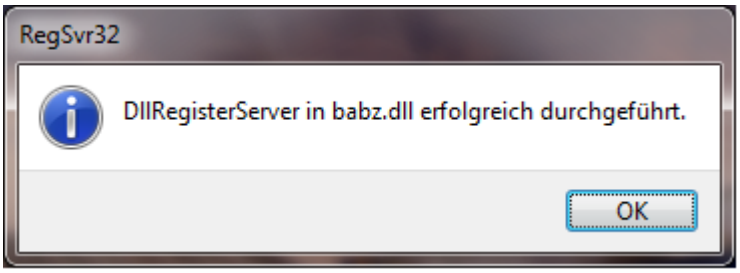

Öffnen
Als Administrator ausführen
Dateipfad öffnen
7-Zip
Scannen mit Microsoft Security Essentials...
An Taskleiste anheften
An Startmenü anheften
Vorgängerversionen wiederherstellen
Senden an
Ausschneiden
Kopieren
Löschen
Umbenennen
Eigenschaften
Wartung
Windows Live
Zubehör
Audiorecorder
Ausführen
Editor
Eingabeaufforderung
Erste Schritte
Kurznotizen
Paint
Rechner
Remotedesktopverbindung
Snipping Tool
Synchronisierungscenter
Verbindung mit einem Projektor
Windows-Explorer
WordPad
Erleichterte Bedienung
Systemprogramme
Windows PowerShell
Zurück
Programme/Dateien durchsuchen
Musik
Spiele
Computer
Systemsteuerung
Geräte und Drucker
Standardprogramme
Hilfe und Support
Herunterfahren

3. Durchführung des Registrierung

Wechseln Sie in der Eingabeaufforderung in den Ordner, in den Sie die Datei „babz.dll“ kopiert haben (verwenden Sie dazu den Befehl „cd“. Im oben gewählten Beispiel geben Sie „cd \babz“ ein und bestätigen Sie mit <Enter>). Geben Sie nun den Befehl „regsvr32 babz.dll“ ein und bestätigen Sie mit <Enter>.

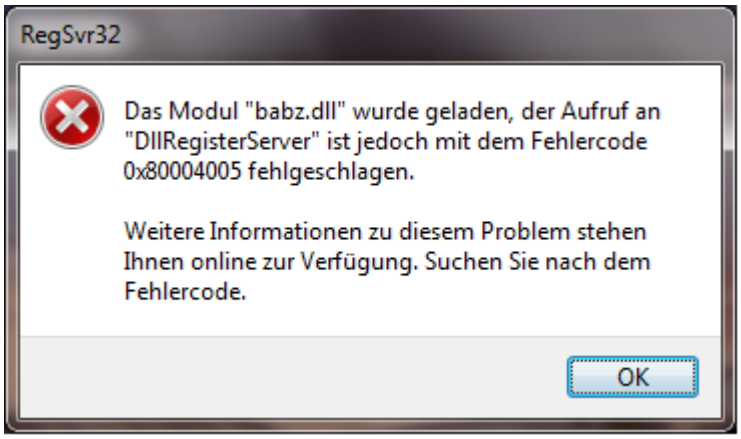


Sie sollten nun die folgende Meldung erhalten:



Die Registrierung wurde erfolgreich abgeschlossen und der babz-Rechenkern kann verwendet werden (z.B. mit dem Interface für Microsoft Excel®).

Wenn Sie nicht über Administratorenrechte verfügen, erhalten Sie die folgende Fehlermeldung. Die Registrierung wurde dann nicht korrekt durchgeführt und babz kann nicht verwendet werden. Stellen Sie sicher, dass Sie die Eingabeaufforderung über „Als Administrator ausführen“ gestartet haben bzw. Ihr Benutzerkonto mit den erforderlichen Rechten ausgestattet ist.


RegSvr32
Das Modul "babz.dll" wurde geladen, der Aufruf an "DllRegisterServer" ist jedoch mit dem Fehlercode 0x80004005 fehlgeschlagen.
Weitere Informationen zu diesem Problem stehen Ihnen online zur Verfügung. Suchen Sie nach dem Fehlercode.
OK

Hinweis zur Aufhebung der Registrierung

Wenn Sie babz von Ihrem System entfernen möchten oder eine neuere Version des Rechenkerns installieren möchten, müssen Sie die Registrierung zunächst wieder aufheben. Gehen Sie dabei genau wie bei der Registrierung vor, verwenden Sie als Befehl jedoch „regsvr32 **-u** babz.dll“.

Anhang III Nutzungsbedingungen

(1) Grundsätzliches

Der Autor (siehe 'Kontakt') stellt dem Anwender eine Kopie des Softwareprodukts 'babz' zur zeitlich befristeten Nutzung zur Verfügung. Mit dem Herunterladen des Softwareprodukts von der Internetseite des Autors erklärt sich der Anwender mit den nachfolgenden Bestimmungen einverstanden. Das Einverständnis ist vor dem Herunterladen zu bestätigen. Ist der Anwender mit den Nutzungsbedingungen nicht einverstanden, so darf das Softwareprodukt nicht heruntergeladen werden.

(2) Aus welchen Bestandteilen besteht das Softwareprodukt?

Das Softwareprodukt umfasst alle vom Autor bereitgestellten Computerdateien, Dokumente und Unterlagen, insbesondere das Hauptprogramm (babz.dll), das Programmbeispiel für Microsoft Excel®, Handbücher und den Lizenzschlüssel.

(3) Erwerbe ich das Softwareprodukt oder das Recht zur Nutzung der Softwareprodukts?

Das Urheberrecht am Softwareprodukt verbleibt beim Autor. Der Anwender erhält lediglich eine zeitlich befristete Lizenz zur Nutzung des Softwareprodukts.

(4) In welchem Umfang darf das Softwareprodukt verwendet werden?

Die Nutzung des Softwareprodukts darf nur bis zum Ablauf der Lizenzdauer oder bis zur Rückgabe der Lizenz erfolgen, je nachdem, welches Ereignis früher eintritt. Nutzungsberechtigter Anwender ist die in der Lizenz genannte natürliche oder juristische Person. Die Lizenz erlaubt die gleichzeitige Installation des Softwareprodukts auf zwei Endgeräten. Die Installation des Softwareprodukts auf einem Server, der die Nutzung auf mehr als zwei Endgeräten erlaubt, insbesondere die Nutzung in einem Intranet oder im Internet, ist nicht gestattet. Nach Ablauf oder Rückgabe der Lizenz sind sämtliche Kopien des Softwareprodukts zu löschen.

(5) Was ist ein Lizenzschlüssel und wie ist mit ihm umzugehen?

Mit jeder Lizenz ist ein Lizenzschlüssel verbunden. Der Lizenzschlüssel enthält Merkmale wie den Namen des Lizenznehmers sowie den Ablauf der Gültigkeit der Lizenz. Der Lizenzschlüssel dient der Freischaltung des Softwareprodukts. Ohne gültigen Lizenzschlüssel kann das Softwareprodukt nicht oder nur eingeschränkt verwendet werden. Der Lizenzschlüssel ist vom Anwender so zu verwahren, dass er Dritten nicht zugänglich ist. Es ist nicht gestattet, Lizenzschlüssel zu verändern sowie Versuche zur Entschlüsselung des Lizenzschlüssels vorzunehmen.

(6) Erhalte ich Updates und an wen kann ich mich bei Fragen wenden?

Mit der Lizenz ist kein Anspruch auf Weiterentwicklung des Softwareprodukts oder Unterstützung des Anwenders bei Installation, Betrieb und Anwendung („Support“) verbunden. Es liegt im Ermessen des Autors, zukünftige Versionen für Inhaber einer Lizenz einer Vorgängerversion zugänglich zu machen. Der Autor bemüht sich, bei entsprechender Nachfrage geeignete Instrumente zum Informationsaustausch zwischen verschiedenen Anwendern bereitzustellen.

(7) Darf ich das Softwareprodukt weitergeben oder im Internet zum Download anbieten? Darf ich meine Lizenz verkaufen, verleihen oder vermieten?

Die Weitergabe des Softwareprodukts (z.B. Verteilung von Datenträgern; Anbieten einer Möglichkeit zum Herunterladen des Softwareprodukts) ist ohne schriftliche Zustimmung des Autors nicht gestattet. Weitergegeben werden darf lediglich das Programmbeispiel für Microsoft Excel®, sofern und soweit dem keine anderen Urheberrechte als diejenigen des Autors, keine Gesetze oder sonstigen Vorschriften entgegenstehen, dem Empfänger kenntlich gemacht wird, dass die Datei von der vom Autor ausgelieferten Datei abweicht bzw. abweichen könnte und die Weitergabe unentgeltlich erfolgt. Der Anwender darf seine Lizenz nicht an Dritte veräußern, verleihen oder vermieten.

(8) Darf ich das Softwareprodukt verändern?

Die Veränderung, Rückentwicklung und Dekompilierung des Softwareprodukts ist nicht gestattet. Ausgenommen ist das Programmbeispiel für Microsoft Excel®, das den Anforderungen des Anwenders angepasst werden kann, sofern die Vorschriften des (7) Satz 2 eingehalten werden.

(9) Was muss ich bei der Verwendung des Softwareprodukts und von mit dem Softwareprodukt angefertigten Berechnungsergebnissen beachten?

Das Softwareprodukt ist nicht für den professionellen Einsatz bestimmt. Mit dem Softwareprodukt erstellte Berechnungsergebnisse dürfen nicht zur Bestimmung von Handelspreisen oder zur Begründung von Handelsentscheidungen herangezogen werden. Mit dem Softwareprodukt ermittelte Berechnungsergebnisse dürfen nicht als Beleg eines Marktpreises, Auflösungspreises oder einer sonstigen Kennzahl, Bestands- oder Bewegungsgröße verwendet werden. Mit dem Softwareprodukt erstellte Berechnungsergebnisse können, bei identischen Parametern und Marktdaten, von den Ergebnissen anderer Computerprogramme und von den Berechnungsergebnissen verschiedener Marktteilnehmer abweichen. Die dem Softwareprodukt zugrunde gelegten Berechnungsvorschriften (mathematische Formeln, Algorithmen) können von den Berechnungsvorschriften anderer Computerprogramme, verschiedener Marktteilnehmer sowie von an den Märkten allgemein anerkannten Berechnungsvorschriften (Usancen) abweichen. Die im Softwareprodukt programmierten Berechnungsvorschriften können von den in der Dokumentation beschriebenen Berechnungsvorschriften abweichen, soweit Änderungen am Softwareprodukt vorgenommen wurden, die noch keinen Eingang in die Dokumentation gefunden haben. Im Rahmen der Entwicklung des Softwareprodukts können Fehler bei der Formulierung und Programmierung von Berechnungsvorschriften unterlaufen sein, wodurch Berechnungsergebnisse fehlerhaft sein könnten. Während des Betriebs des Softwareprodukts können Fehler auftreten, wodurch Berechnungsergebnisse verfälscht werden könnten (Laufzeitfehler).

(10) In welchem Umfang haftet der Autor, wenn mir durch das Softwareprodukt ein Schaden entsteht?

Das Softwareprodukt wurde mit großer Sorgfalt entwickelt, jedoch können Fehler aufgrund der Komplexität des Softwareprodukts niemals ganz ausgeschlossen werden. Zum Schutz vor Schadprogrammen (Viren) setzt der Autor aktuelle Virenschutzprogramme ein, dennoch könnten Bestandteile des Softwareprodukts von Viren befallen sein oder im Zuge der Bereitstellung an den Anwender von Viren befallen werden. Der Autor schließt, soweit gesetzlich zulässig, jede Haftung für sich aus dem Softwareprodukt sowie aus

mit dem Softwareprodukt übertragenen Viren ergebenden Schäden (z.B.: Schäden an Software und Hardware; Verlust von Daten; Schäden aus fehlerhaften Berechnungsergebnissen; entgangener Gewinn; Geschäftsunterbrechung; Unmöglichkeit der Erfüllung einer Pflicht) aus.

(11) Anwendbares Recht

Es gilt das Recht der Bundesrepublik Deutschland.

Anhang IV Historie

Folgende Änderungen und Neuerungen wurden gegenüber der Vorgängerversion umgesetzt:

1. **Berechnung des Nominalzinssatzes aus der Zinszahlung**

   Bei Objekten der Klasse BALoan können nun für die manuelle Zinssteuerung nicht nur Nominalzinssätze angegeben werden, sondern alternativ Zinszahlungen, aus denen dann der Nominalzinssatz abgeleitet wird. Weitere Hinweise finden Sie im Abschnitt „7.3.11 ManualInterest“ ab Seite 94.

2. **Definition von Ausnahmeregeln für die Feiertagsermittlung**

   Für bestimmte Feiertagstypen können Objekte der Klasse BACalendar Ausnahmeregeln hinterlegt werden. Dadurch können Feiertage in einzelnen Kalenderjahren gestrichen oder verschoben werden. Außerdem ist es möglich, Feiertage nur in bestimmten Kalenderjahren anzuwenden. Für vordefinierte Feiertage können keine Ausnahmeregeln definiert werden. Weitere Hinweise finden Sie im Abschnitt „3.2.3 Holidays“ ab Seite 29.

3. **Änderung der Nutzungsbedingungen**

   Nutzungslizenzen und die zugehörigen Lizenzschlüssel werden derzeit ausschließlich kostenlos vergeben. Die Regelungen zur Erstattung des Kaufpreises bei Widerruf wurden daher aus den Nutzungsbedingungen gestrichen.

4. **Anpassung des Akronyms „babz“**

   Aus zeitlichen Gründen ist es nicht wahrscheinlich, dass babz in näherer Zukunft umfangreichere Funktionalitäten zur Berechnung von Kennzahlen erhält. Im Vordergrund wird daher weiterhin der Aufbau und die Bewertung von Zahlungsströmen stehen, nicht die Analyse und die Bewertung. „babz“ steht daher ab sofort für „Bibliothek zum Aufbau und zur Bewertung von Zahlungsströmen“.

Diese Auflistung muss nicht alle Änderungen widerspiegeln.